2019

# Application of long-short term memory Deep Neural Network in financial forecasting

Peter W. Wanyonyi
*Strathmore Institute of Mathematical Sciences (SIMS)*
*Strathmore University*

# Application of Long-Short Term Memory Deep Neural Network In Financial Forecasting

**Wanyonyi Watua Peter**

**Submitted in partial fulfillment of the requirements for the Masters of Science in Mathematical Finance at Strathmore University**

**Strathmore Institute of Mathematical Sciences**
**Strathmore University**
**Nairobi, Kenya**

**June, 2019**

# Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dessertation contains no material previously published or written by another person except where due reference is made in the thesis itself.



Wanyonyi, Watua Peter

.....

June 1th, 2019.

**Approval**

The dissertation of Wanyonyi, Watua Peter was reviewed and approved by the following:

Dr. Lucy Muthoni,
Lecturer- Strathmore Institute of Mathematical Sciences.
Strathmore University

Ferdinand Othieno,
Dean, Strathmore Institute of Mathematical Sciences,
Strathmore University

Professor Ruth Kiraka,
Dean, School of Graduate Studies,
Strathmore University

# Abstract

The goal of this research was to apply Long-Short Term Memory Deep Neural Networks in financial forecasting. In order to predict the financial data, we used long-short term model and we compared its performance to ARIMA-GARCH hybrid model. In the study we used the ARIMA-GARCH time series model and studied its limitations in time series forecasting. We then introduced Deep neural network model (DNN) so as to to improve accuracy which was tested on different financial datasets. Lastly we compared the results of the models employed using the root mean square error(RMSE) and p-value; LSTM had RMSE of 0.09989178 while the ARIMA-GARCH had RMSE of 0.0178. it was then concluded that the long-short term memory (LSTM) model, which is one of the DNN models, had significantly better than the ARIMA-GARCH Hybrid model in prediction/forecasting financial returns on FTSE100 and S&P500 indices.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

DNN Deep Neural Network

CNN Convolutional Neural Networks

LSTM Long-Short Term Memory

RNN Recurrent Neural Networks

ARMA Autoregressive Moving Average

ARIMA Autoregressive Intergrated Moving Average

ARCH Autoregressive Conditional Heteroskedastic

GARCH Generalised Autoregressive Conditional Heterskedastic

DNN Deep Neural Networks

NYSE New York Stocks Exchange

UK United Kingdom

ADAM Adaptive Monument Estimation

RMSE Root Mean Square Error

API Application Programming Interface

SVM Support Vector Machines

# Acknowledgement

I would like to convey my deepest sense of gratitude to my supervisor Dr. Lucy Muthoni, for her inspiration and her invaluable suggestions. I am indebted to her for giving much support while I was unsure that I would be able to complete and I learned and gained so much in the process.

I acknowledge with thanks, the assistance provided by the Department staff and all the members of the School of Mathematical Sciences. Finally, I would like to extend my deepest humble gratitude to my classmates who directly or indirectly helped me for the same.

# Dedication

This is for my father Patrick Wanyonyi, my mother Ketty Wanyonyi, my love Kelly, my daughter Ameerah Watua, my brothers and sisters. Thank you so much for the encouragement and being there for me.

# Chapter One

# 1 Introduction

## 1.1 Background of the Study

Time series is defined as sequence of data points that have natural but temporal occurring data points collected sequentially over a span of time, the time frame can be discrete or continuous. Time series series analysis of data more specifically in the area of finance is about inferring what happened to a series of data points in the past and trying to predict what will happen to it in the future, (Halls-Moore, 2017). When carrying out time series analysis we are either faced with the decisions that require segmentation, curve fitting, prediction and forecasting, classification, function approximation and/or clustering categories (Hatami, Gavet, & Debayle, 2018). Generally, time series data set often contain trends, which are consistent directional movement in time series; seasonal variation which are normally seen in commodities and serial dependence/autocorrelation also known as serial correlation which is brings about volatility clustering.

Volatility clustering is a feature of serial correlation that is of interest when it comes to time series analysis in areas of econometric. These features of time series are complicated to analyze and forecast precision because of non-linear trends, noise present in the time series and heavy tails (Cont, 2001).

According to (Borovykh, Bohte, & Oosterlee, 2017), when building models for forecasting and prediction of financial market data, it is desirable that the model captures the dependencies in the data as well as have resilience to noise. Unfortunately, time series analysis of the current markets are marred by big data, that the traditional autoregressive models such as ARIMA family of models and GARCH family of models fail to capture the above features exhaustively. In addition to that they are also prone to overfitting, (Hansen & Nelson, 1997). Hansen also explains that financial time series data is becoming complex due to different regimes that are exhibited in different markets. These challenges can be addressed by applying other sets of models which can adequately compliment the traditional models.

## 1.2 Deep Neural Networks

Traditional models can be complemented by deep neural networks models (DNN). DNN models belong to a class of machine learning models that are founded on learning data representations, which differs to the commonly known task specific algorithms. They have been a sought after way of learning the structural dependencies in the data because of their abilities to learn linear and non-linearities without the need of specifying a particular model form in advance (Zhang, Patuwo, & Hu, 1998).

Applications of deep neural networks in time series analysis was initially done by (Gamboa, 2017), who used DNN to carry out time series analysis by transforming anomalies into time series through finding regions where values are too different from the actual ones. The model has since been used in finance and econometrics for example in market predictions(Dixon, Klabjan, & Bang, 2017) and leveraging high frequency trading (Arévalo, Niño, Hernández, & Sandoval, 2016). The widely used deep neural network model in finance has been convolutional neural networks, which

previously had a huge success in image classification sector.

The main advantage of Deep neural networks is the fact that they require vast sample of data for one to obtain a stable prediction outcome. They can also be applied in both supervised learning tasks and unsupervised learning tasks where there is vast amount of unlabeled datasets. In this study we are going to use the long-short term memory (LSTM) to analyze financial time series data.

## 1.3 Problem Statement

The amount of time series data generated by financial markets has been on the rise in the recent years. It is also expected that the complexity of data will keep on growing and therefore determination of various variables present in the market will become harder and harder. In addition to that there has been observed increasing systematic and unsystematic factors in the financial markets. Analyzing data in such complex systems requires more sophisticated tools than are currently used in the markets. In forecasting of financial variables for instance, it has been shown that the traditional methods are no longer adequate and therefore, a need to apply DNN in time series data, (Gamboa, 2017).

This study seeks to compare the fitting of traditional models on time series data with that one of deep neural network models.

## 1.4 Objectives of the study

The main objective if this study is to determine which is the best forecasting method for financial data between ARIMA-GARCH and DNN. The specific objectives are:

a. establish the best traditional forecasting method for financial data

b. Compare deep neural networks with traditional method and determine which is superior

## 1.5 Research Questions

We seek to find answers to the following questions:

a. Among the traditional time series forecasting models, which is the best?

b. Between the best traditional time series forecasting model and DNN which is superior in forecasting financial data?

## 1.6 Significance of the Study

The results of this work will be used to guide the financial experts and traders on the best time series predicting tool. The results of the study will also contribute to different sectors such as agriculture, energy and real estate which have constant accumulation of massive data and needs solving of different time series problems.

# Chapter Two

## 2 Literature Review

### 2.1 Introduction

Immense development have been done in the forecasting of time series data over a span of time. In recent years, researchers have continuously developed sophisticated forecasting models in an attempt to asses the features of time series data. This chapter seeks to evaluate the literature on the models that have been developed in the past that have been used in financial forecasting. Section 2.2 explains the current models that are being used for financial forecasting. Section 2.3 reviews literature on volatility forecasting, section 2.4 reviews the comparison between deep neural networks and traditional Methods and section 2.4 summarizes the research gap.

### 2.2 Current Financial Forecasting Models

Different models have been developed for financial forecasting.(L. Cao & Tay, 2001) performed time series forecasting using Support Vector Machines(SVM), in their studies they compared financial data prediction/forecasting by differentiating it with a multi-layer perceptron trained using the Back Propagation algorithm. In their studies they found out that SVMs have better forecasting ability than back propagation based on the criteria of Mean Absolute Error, Normalized Mean Square Error, Directional Symmetry, Correct Up trend and Correct Down Trend. They used the model on S&P daily price index.(Kim, 2003) also conducted similar study, the study used SVM to forecast the direction of the future stock price index. They investigated the effect of the value of the upper bound C and the Kernel parameter in SVM. The results showed that the prediction performances of the SVMs respond sensitively to the value of the above parameters. Thus, it is critical to find the supreme value of the parameters.

Financial predictions using genetic algorithms was inspired by (Mahfoud & Mani, 1996), in which the algorithm was applied to about 5000 individual stocks for purposes of predicting futures performances. The genetic algorithm was benchmarked against a neural network system, in their experiments they studied that both systems significantly outperformed 'the market,' with the genetic algorithm producing significant results. They also combined both algorithms and they found out that the combination outperformed either algorithm individually.

(Selvin, Vinayakumar, Gopalakrishnan, Menon, & Soman, 2017) conducted a study in which they used some of the models that are used in this study, they tried to predict the stock index movement for a single company using the end of day closing prices. However, the focus was not to fit data to specific model, they identified latent dynamics existing in the data using deep neural networks, they used RNN, LSTM, and CNN and the quantified the models using percentage error. The study showed that DNN are able to capture hidden dynamic and are suitable for forecasting since they are able to identifying the inter relation within data. The study further showed that the CNN architecture was capable of identifying trend changes. From their methodology, they showed that CNN model was the best. They highlighted the fact that changes in the stock market may not be in an unvarying pattern. This study forecast more on implementing the CNN rather than incorporating all of the models that were mentioned. Similar studies was carried out by (Dingli & Fournier, 2017) In the study they focused on CNNs to forecast the next period price direction with respect to the current price. An accuracy of 65% when forecasting was achieved on the next months price and 60% accuracy for the next week price direction forecast. They failed to outperform the results obtained by other compared models, such as Logistic Regression and SVMs. The approach they used argued that there is no randomness in the stock prices.

In the study by (Doering, Fairbank, & Markose, 2017), they applied CNN to high frequency market-microstucture forecasting, in the study deep convolutional neural network was trained on market microstructure data from London Stocks Exchange, for forecasting purposes. It was found that the deep-learning models used when combined with developed data transformation, performed well. The CNN was used because of the infinite number of layers that you can used in order to increase the accuracy.

Financial time series forecasting using SVMs was extended by (L.-J. Cao & Tay, 2003), in the study the researcher included the adaptive parameters in the forecasting. They argued that SVM does not have significant results in forecasting unless they are supported by other parameters. Practicability of applying SVM in financial time series forecasting was examined by comparing it with multi-layer back propagation neural network and the regularized radial basis function neural network. The adaptive parameters were then proposed by incorporating the nonstationary of financial time series into the SVM. They used future data fro the Chicago Mercantile market. In conclusion they showed that SVM outperforms back propagation neural network in financial forecasting. The free parameters of SVM had a great effect on the generalization performance. The SVM were used so that they could minimize an upper bound of the generalization error rather than minimizing the training error.

## 2.3 Volatility Forecasting

Traditionally GARCH family models have been used used in volatility forecasting, different models have been developed to asses volatility (Crawford & Fratantoni, 2003) analyzed the forecastability of the house prices using ARIMA and GARCH and regime-switching. They added regime switching models because series of house prices behaves differently depending on the realization of an unobserved regime variable and the are the near great choice for real estate markets. Thus they were used in-sample while ARIMA models were superior in out-sample forecasting. The study compared the in- and out-of-sample forecasting performances on the three models. They concluded that regime-switching models performed better in-sample, while simple ARIMA models showed significant performance in out-of-sample forecasting.

(Xiao, Xiao, Liu, & Wang, 2014) used a different approach, in the study they incorporated ARIMA and (artificial neural networks) ANN for financial market volatility forecasting. In the study, the series was first decomposed to different scale by maximum overlap discrete wavelet transform. The approximation was done by combining ARIMA and feedforward neural network(FNN). ARIMA was used to generate linear forecast while the feedforward neural networks was used to capture non linear part. This was one of the basis for the study, we chose to apply the GARCH model instead to take care of the non linear part. (Rathnayaka, Seneviratna, Jianguo, & Arumawadu, 2015) extended this study by combining the ARIMA and artificial neural networks (ANN) for time series forecasting. The study aimed at forecasting stock prices under high volatility, this was an attempt to understand the behavioural patterns and to develop forcasting approach based on ARIMA-ANN for estimating the price index using data from Colombo Stocks Exchange. The study showed that ARIMA(4,1,3) and ARIMA(1,1,1) traditional approaches were suitable for predicting the ASPI and SL20 price indexes but model testing results of Mean absolute deviation showed that ARIMA-ANN approach was the most suitable for price indices under the high volatility than other traditional forecasting methods.

(Yaziz, Azizan, Zakaria, & Ahmad, 2013) combined ARIMA and GARCH to form a hybrid that was used to forecast the gold price. The best ARIMA model was used to model the linear data of time series and the residual of the this linear model will contain the nonlinear error term, GARCH was used to model the nonlinear patterns of the residuals. The empirical results concluded that ARIMA-GARCH model improved in forecasting accuracy by five fold compared to the previously applied forecasting methods. The study concluded that combination of ARIMA which offer flexibility and power, and GARCH which offer strength in modeling volatility and risk in time series have the potential to overcome the limitations in time series data.

## 2.4 Comparison between Deep neural Networks and Traditional Methods

previous scholars have been able to compare the performance of the traditional methods versus the deep neural networks methods. (Hansen & Nelson, 1997), compared the neural networks and the traditional time series methods. They applied neural networks in forecasting tax revenues in United States, the study was leveraging the motif finding capabilities of neural networks to give alternative views of the seasonal and cynical components found in the financial markets time series.

The study highlighted that the applications stated in the above example in the revenue forecasting demonstrated how deep neural networks complimented the traditional models, for example, preoccupation with potential downturn in the economy distracts analysis that are based on traditional time series models such that it overlooks an emerging new phenomenon in the data. In such a case, DNN identify the new pattern that then allows modification of the time series models and thus gives more superior results. They further highlighted that the data structures found by traditional statistical tools enables an analyst to provide the DNN with important information that the networks to create more accurate models. The study concluded that the synergy between the two models resulted in a portfolio of forecasts being more accurate than the total sum of the individual parts.

The study carried out by (Dasgupta, Dispensa, & Ghose, 1994), compared the predictive performance of neural network model with traditional models showed that neural network models perform better that the traditional models. However, the study highlighted that the difference in performance was not significant.

The paper, (Adhikari, Agrawal, & Kant, 2013) also compared the performance of ANN models, feedforward models and Elman artificial neural networks against SARIMA and Support Vector Machine models. The study concluded that the neural network models achieved reasonably better forecasts compares traditional models.

## 2.5 Research Gap

In the study by (L. Cao & Tay, 2001) in which they used SVMs in time series forecasting, they concluded that the SVMs performed better in time series analysis. In another study by (L.-J. Cao & Tay, 2003), they concluded that SVMs need to be combined with adaptive parameters for them to have a significant performance in financial time series. (Dasgupta et al., 1994), opined that application of the artificial neural networks in forecasting, was significant compared to that of SVM and SARIMA models.

(Crawford & Fratantoni, 2003) in their study that they applied GARCH and ARIMA models, they concluded that these models had significant performance in forecasting the prices of house. The same study was conducted by (Yaziz et al., 2013), in the study they used ARIMA and GARCH models to form a hybrid model. They used the model to forecast the price of gold, which they concluded that the model performed 5 times better than the models they were comparing the performance with.

From the above studies, we are going to revaluate the performance of ARIMA-GARCH traditional models compare it to DNN architecture.

# Chapter Three

## 3 Methods and Materials

### 3.1 Introduction

The chapter expounds on the methodology that was used to achieve the research objectives. This chapter will highlight the target population, research design, sampling design, data collection and analysis methodologies. The study develops a model that examines the performance of deep neural networks in financial forecasting.

### 3.2 Research Design

This study is explanatory given that little research has been carried out on application of long-short term memory deep neural networks in financial forecasting using market data. It analyses the pattern structures in the real datasets and try to explain the serial correlations in the data.

### 3.3 Sampling

This research took a test and train approach where the data was divided into two random samples, the training data set and the test data set. A cross-sectional validation approach will then be implemented during the training and testing of the model, and the model performance analysis.

The time series data used for this study was obtained from Algoseek (www.Algoseek.com accessed 3rd March 2019), and yahoo finance (www.yahoo.com, 3rd March 2019). It includes the historical information on the closing prices of the indices. The data was then used in forecasting in the traditional model and also to train and test the LSTM model.

### 3.4 Data Collection

The data that was used in this study was secondary data. The data was extracted from Algoseek, which is second tier data vendors that deal in historical financial data. The study period is between

2007 - 2019; we use massive data because the machine learning models require a lot of data for them to be trained efficiently.

## 3.5 Model Framework

Section 3.5.1 borrows from (Yaziz et al., 2013) in building the ARIMA-GARCH model in developing the forecasting models, the section further explains the model in details. Section 3.10 will focus on deep neural network and explain the long-short term model (LSTM) which we are going to use for comparison and final forecasting.

## 3.6 Models

### 3.6.1 Autoregressive Moving Average (ARMA) prototype of order p,q

A timeseries model, $x_t$, is said to be an autoregressive moving average model of order p,q, ARMA(p,q), if:

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + ... + \omega_t + \beta_1 \omega_{t-1} + \beta_2 \omega t - 2... + \beta_q \omega_{t-q} \tag{3.6.1}$$

Where $\omega_t$ is called a discrete white noise process with $E\omega_t = 0$ and variance $\sigma^2$. If we consider the Backward Shift Operator, **O** then we can rewrite the above as a function of $\theta$ and $\phi$ of **O**:

$$\theta_p \mathbf{O} x_t = \phi_p(\mathbf{O}\omega_t) \tag{3.6.2}$$

By setting $p \neq 0$ and $q \neq 0$ we shall recover MA(q) model.

One of the main feature of ARMA model is that it is *parsimonious* and *expendable* in its parameters (Halls-Moore, 2017). By this we mean, an ARMA model often need less parameters than an AR(p) or MA(q)model alone.Additionally, if we rewrote the equation in terms of the backward shift operator then $\theta$ and $\phi$ polynomials sometimes share a common factor hence leading to a more simpler model (Halls-Moore, 2017).As studies have shown, ARMA models are never superior fits for log returns of equities.

### 3.6.2 Autoregressive Integrated Moving Average and Conditional Heteroskedastic Models

In this section we discussed an addition of ARMA model, in full the Autoregressive Integrated Moving Average model as well as the models that incorporate conditional heteroskedasticity, such as ARCH and GARCH.

### 3.6.3 Autoregressive Integrated Moving Average (ARIMA) Models of order p,d,q

ARIMA model is made by joining autoregressive (AR) and the moving average (MA) models both of which are univariate time series models. The model makes use of the past information of a time series to predict the future values for the series. The ARIMA model is of order of ARIMA(p,d,q) where $p, d$ and $q$ are integers that falls on the range zero or greater than zero. In the model, $p$ gives the number of autoregressive lags, $d$ is the order of integration that makes the data stationary, while $q$ is the number of lags in the moving average(MA). ARIMA models are often great to be considered whenever we are dealing with non-stationery series, such a series occur in the presence of stochastic trends (Halls-Moore, 2017). This is because ARIMA models can lessen a non-stationery series to a stationary series using a sequence of differencing steps(Hamilton, 1994). If we apply the difference operator to a random walk series $x_t$ (a non-stationery series) we remain with a white noise $\omega_t$(a stationary series):

$$\nabla x_t = x_t - x_{t-1} = \omega_t \tag{3.6.3}$$

ARIMA essentially accomplishes this function but does so repeatedly d times so that it can reduce a non-stationery series to a stationery one. Consider an **Integrated Series of order d**. A time series $x_t$ is *integrated of order d, I(d)*, if:

$$\nabla^d x_t = \omega_t \tag{3.6.4}$$

This implies that if difference the series $d$ times we remain with a discrete white noise series. It is also possible to use the Backward Shift Operator **B** to provide an equivalent condition:

$$(1 - \mathbf{B}^d)x_t = \omega_t \tag{3.6.5}$$

Thus, given the above definitions we can define the ARIMA process as a model of order p, d, q, **ARIMA (p,d,q)**, if $\nabla^d x_t$ ARMA model of order p,q written as ARMA(p,q) (Halls-Moore, 2017). The implication of this is that, if the series $x_t$ is differenced $d$ times, and it then follows an ARMA(p,q) process, then it is an ARIMA(p,d,q) series.If we use the polynomial notation used in ARMA above then ARIMA(p,d,q) process can also be written in terms of the Backward Shift Operator, **O** (Halls-Moore, 2017):

$$\theta_p(\mathbf{O})(1-\mathbf{O})^d x_t = \phi_q(\mathbf{O})\omega_t \tag{3.6.6}$$

where $_t$ is a discrete white noise series.

If we suspect presence of a non-linear trend then we might be able to use repeated differencing (i.e. $d > 1$) to diminish a series to stationery white noise.

## 3.7 Conditional Heteroskedasticity Models

Given a set of random variables, such as elements in a time series model, we say the collection is *heteroskedastic* if there exists a certain subsets, of within the larger set that have a different *variance* from the remaining variables(Nelson, 1991). A series that experiences Heteroskedasticity that is serially correlated and hence conditional on periods of increased variance are known as **conditional heteroskedastic**.

One of the challenges of conditional heteroskedastic series is that if we plot the correlogram of a series with volatility then we might still see what appears to be a realization of *stationary* discrete white noise(Nelson, 1991). The volatility is flinty to detect purely from the correlogram. The following models are used for detecting conditional heteroskedastic series.

### 3.7.1 Autoregressive Conditional Heteroskedastic (ARCH)Models

Autoregressive Conditional heteroskedastic (ARCH) model of order unity is defined as follows: A time series $\epsilon_t$ is given at each instance by:

$$\epsilon_t = \sigma_t \omega_t \tag{3.7.1}$$

Where $\omega_t$ is a white noise, with a of mean zero and variance of one, and ${\sigma_t}^2$ is given by:

$$ {\sigma_t}^2 = \alpha_0 + \alpha_1 {\epsilon_{t-1}}^2 \tag{3.7.2} $$

Where $\epsilon_0$ and $\alpha_1$ are parameters of the model. $\epsilon_t$ is an *autoregressive regressive conditional heteroskedastic model of order unity*, this is denoted by ARCH(1). Substituting for ${\sigma_t}^2$, we get:

$$ \epsilon_t = \omega_t \sqrt{\alpha_0 + \alpha_1 {\epsilon_{t-1}}^2} \tag{3.7.3} $$

When fitting an AR(1) model we focused on the decay of the first lag on a correlogram of the series. Nevertheless, if we apply the same rationale to the *square* of the residuals and see whether we can apply an AR(1) to the squared residuals, then we shall have an indicator that an ARCH(1) process maybe suitable (Halls-Moore, 2017).

ARCH(1) applies to series that already has an appropriate model fitted sufficient to leave the residuals that replicates a white noise. We can only confirm whether ARCH is appropriate or not by squaring the residuals and examining the correlogram. We note that, ARCH should only be applied to series that don't have any trends or seasonality effects, which are those that have no evidence of autocorrelation (Halls-Moore, 2017).

### 3.8 Generalised Autoregressive Conditional Heteroskedastic, GARCH Models

Generalised Autoregressive Conditional Heteroskedastic Model is an extension of ARCH model which is very similar in nature to the addition of an AR to ARMA process. $GARCH(p, q)$ model has the same equation as $ARCH(1, 1)$ for the log returns the difference being the variance (volatility) term. GARCH model of Order p,q is defined as: A series $\epsilon_t$ that is given at each instance by:

$$ \omega_t = \sigma_t \omega_t \tag{3.8.1} $$

Where, $\omega_t \sim (0, 1)$, that is, it is a white noise and $\sigma_t^2$ is given by:

$$\sigma_t^2 = \alpha_0 + \sum_{j=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \tag{3.8.2}$$

Where $\alpha_i$ and $\beta_j$ are the boundaries of the model. $\epsilon$ is a generalised autoregressive conditional heteroskedastic model of the order p,q, denoted by GARCH(p,q). If we let the $r_t$ be the mean corrected return, $\epsilon_t$ be the discrete white noise as defined before and $H_t$ be the information set at a particular time $t$ given by $H_t = r_1, r_2, ....., r_{t-1}$ we can write GARCH model as:

$$r_t = \sigma_t \epsilon_t \tag{3.8.3}$$

$$\sigma^2 = \alpha_0 + \alpha_1 r_{t-1}^2 + \beta_1 r_{t-1}^2 \tag{3.8.4}$$

In forecasting using GARCH we consider a forecasting origin of time $T$; then the $l-$ step ahead volatility forecast will be:

$$r_t^2(l) = E[r_{t+1}^2 | r_t] \tag{3.8.5}$$

$$\alpha_0 + \sum_{i=1}^{p} (\alpha_i + \beta_j) E(r_{t+l-i}^2 | r_t) - \beta_j \sum_{i=1}^{q} E(v_{t+l-i|r_t} \tag{3.8.6}$$

Where $r_t^2, ...., r_{t+l-m}^2$ and $\sigma_t^2, ....., \sigma_{r+l-p}^2$ are assumed to be known at time t and the true parameter values $\alpha_i$ and $\beta_j$ for $i = 1...m$are then replaced by their estimates. $E[r_{t+1}^2 | r_t]$ for $i < l$ can be given recursively as for $l >= i, E(v_{t+l-1} | r_t = 0)$ for $i < l, E(v_{t+l-i|v_t+l-i}$ for $i >= l$

### 3.9 ARIMA-GARCH Models

We opted for an ARIMA-GARCH model as the choice of traditional models instead because the for GARCH model the normal distribution does not completely explain the skewness and leptokurtosis of

the financial time series. Thus, the need for an ARIMA-GARCH model adds another assumption on the z-distribution. Studies show that when ARIMA model is used alone only predicts the conditional mean (returns) while the GARCH model used alone only forecast the conditional volatility. When both models are combined to form n ARIMA-GARCH hybrid, they forecast returns that are adjusted to volatility.

Consider a series $y_t$ that has unconditional mean zero and follows an $ARMA(p,q)-GARCH(s,r)$ process. An $ARMA-GARCH$ hybrid model is defined as:

$$y_t \sim D(\mu_t, \sigma_t^2) \tag{3.9.1}$$

where

$$\mu_t = \varphi_1\mu_{t-1} + ... + \varphi_p\mu_{t-p} + (\varphi_1 + \theta_1)\mu_{t-1} + ... + (\varphi_n + \theta_n)\mu_{t-m} \tag{3.9.2}$$

and

$$\sigma_t^2 = \omega + \alpha_1\mu\mu_{t-1}^2 + ... + \alpha_s\mu_{t-s}^2 + \beta_1\sigma_{t-1}^2 + ... + \beta_r\sigma_{t-r}^2 \tag{3.9.3}$$

and

$$\frac{\mu_t}{\sigma_t} \sim i.i.D(0,1) \tag{3.9.4}$$

Where $\mu_t = y_t - \mu_t$ $D$ is assumed to be an some density, for instance, normal $\varphi_i = 0$ for $i > p$; and $\theta_j = 0$ for $j > q$ The conditional mean process due to ARMA has the shape as the conditional variance process due to GARCH, the lag orders may differ(allowing for a nonzero unconditional mean of $y_t$). None of the models has random error terms once conditioned on $l_{t-1}$, hence both are implemented. Also in implementation of the model in the research, the hybrid model has two phases, the first one involves using the best of ARIMA models in the linear data of time series and the residual of this linear model will then have non-linear data only. The second part involves using GARCH to

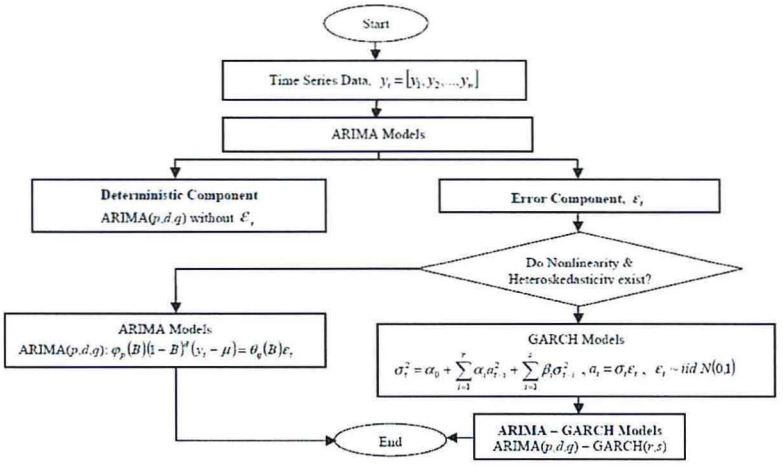


Figure 1: Hybrid Procedure of ARIMA-GARCH models Flow Chart

model the nonlinear patterns of the residuals. The model thus uses the GARCH to model the error components, then the resultant model is applied to analyze the series and predict values, (Tan, Zhang, Wang, & Xu, 2010). In this process, the error term$\epsilon$ of the ARIMA model is assumed to follow a GARCH process of orders $p$ and $q$. This procedure has been summarized in figure 1

## 3.10 Deep Neural Network Architecture

In this section we are going to highlight briefly how the architecture on the neural network is structured and how the network that was used in the study will be structured.

### 3.10.1 Feedforward neural networks

Feed forward neural networks consists of L number of layers with $M_n$ hidden nodes in each of layer $l = 1, ....., L$. Suppose we have an input $x(1), ...., x(n)$ and we want to use the multi-layer neural network to output the forecast value at the next time step $\hat{x}(t+1)$. In the first layer we construct $M_1$ linear combinations of the input variables is of the form

$$a^1(i) = \sum_{j=1}^{t} \omega^1(i,j)x(j) + b^i, for i = 1, ..., M_1 \tag{3.10.1}$$

Where $\omega^1 \in^{M_1 \prod t}$ are the weights and $b^1 \in^{M_1}$ as the biases. Each of the outputs $a^1(i)$, $i = 1, ..., M_1$ are then transformed by a differentiable, non-linear activation function to give

$$f^i = h(a^1(i)); i = 1, ..., M_1 \tag{3.10.2}$$

The role of the non-linear function is to enable the model to learn the non-linear relations between the data points. In every subsequent layer the outputs from the previous layer $f^{l-1}$ are again linearly combined and then passed through the non-linearity.

$$f^l(l) = h(\sum_{j=0}^{M_{l-1}} \omega^L(j) + b^L) \tag{3.10.3}$$

with $\omega_l \in \mathbb{R}^{M_l \times M_{l-1}}$ and $b^l \in \mathbb{R}^M$. In the final layer $l = L$ of the neural network, the forecast value $\hat{x}(t+1)$ is evaluated using

$$\hat{x}(t+1) = h(\sum_{j-0}^{M_{L-1}} \omega^l(i,j) f^{l-1}(j) + b^l(j)); i = 1, ..., M_1 \tag{3.10.4}$$

with $\omega_l \in \mathbb{R}^{M_l \times M_{l-1}}$ and $b^1 \in \mathbb{R}$. In a neural network, every node is thus connected to every node in adjacent layer.

### 3.10.2 The Long-Short Term Memory Networks

LSTM networks are built from Recurrent Neural Networks, they are much more powerful in that they are capable of learning long-term dependencies. Normally in RNN, small weights are multiplied recurrently through several times steps and the gradients diminish asymptotically to zero. This is commonly known as vanishing gradient problem and it is one of the downsides of RNN. In LSTM, the network is made up of memory blocks known as cells that are connected to each other through layers. The information in these cells are contained in the cell state $C_t$ and hidden state $h_t$. This information is regulated by gates through sigmoid and the activation function is $tanh$

Usually, the sigmoid function outputs numbers in the range of 0 and 1, when it indicates 0, this implies that no information is going through, if it indicates 1 then every information is going through and this is the main focus. Therefore, long short term memory network has the capability to delete or add information from the cell state conditionally.

The general idea is, the gates take in the input, then the hidden states from the previous time step $h_{t-1}$ and the current input $x_t$ and multiply them pointwise by weight matrices, $w.$ and bias $b$ is added to the product.The three main gates are as follows:

The forget gate, this determines the information to be deleted from a given cell state, then outputs the number between 0 and 1, with zero meaning delete all and 1 implying remember all. This is given in the equation (3.11.1):

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{3.10.5}$$

The next gate is the input gate, this is made up of $tanh$ activation layer which creates a vector of potential candidate as shown in the following equation:

$$\hat{C}_t = tanh(w_c[h_{t-1}, x_t] + b_c) \tag{3.10.6}$$

This allows the sigmoid layer to create an update filter as follows:

$$U_t = \sigma(w_u[h_{t-1}, x_t] + b_u) \tag{3.10.7}$$

This followed by the update of the old cell state, $C_{t-1}$ to:

$$C_t = f_t * C_{t-1} + U_t \times C_t^* \tag{3.10.8}$$

Finally we have the output gate which has the sigmoid layer that filters the cell state that is going to the output. This is given by the equation:

$$O_t = \sigma(w_0[h_{t-1}, x_t] + b_0) \tag{3.10.9}$$

The cell state $C_t$ is then passed through the $tanh$ function to scale the values to the range of [-1,1]. Lastly, the scaled cell state is then multiplied by the filtered output to obtain the new hidden state,$h_t$ to be passed on to the next cell, this is given by the equation

$$h_t = O_t \times tanh(C_t) \tag{3.10.10}$$

### 3.11 Model Validation

In order to choose between the separate best models. To choose the best ARIMA model we used the Akaike Information Criterion (AIC) which estimates the 'quality' of each model relative to other available models. if we calculate the likelihood function for a given statistical model, which has $k$ parameters, and L is the maximized value of the likelihood function for the estimated model that maximizes the likelihood, then the $AIC$ is given by:

$$AIC = -2log(L) + 2k = 2k - 2nlog(\frac{RSS}{n}) \tag{3.11.1}$$

In the second equation, RSS is the Residual Sum Squares for the estimated model and n is the number of observations. To select the preferred model from the selection of models we choose the model that has the minimum AIC of the group. From the equation we note that the AIC with the number of parameters, $k$, increases but it is reduced if the negative log-likelihood increases. AIC works by penalizing the models that are overfit.

After a model has passed all the diagnostic checks, it becomes suitable for predicting. We created the ARIMA models of varying orders using the AIC. We supplemented the AIC by performing a Ljung-Box test, this is a classical hypothesis test that is developed to test whether a given set of serial correlations of a fitted time series model may differ significantly from zero. However, this test doesn't test each individual lag randomness, but it tests the randomness over a group of lags.

In this test we define the null hypothesis $H_0$ as: The time series data at each lag are independent and identically distributed. We define the alternative hypothesis $H_1$ as: The time series are not independent and identically distributed and possess serial correlation.

Next we calculate the following test statistic, Q:

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k}{n-k} \tag{3.11.2}$$

Where $n$ is the length of the series sample, $\hat{\rho}_k$ is the sample serial correlation at lag $k$ and $h$ is the number of lags under the test.

The decision criteria as to whether we reject the null hypothesis is to check whether $Q > \chi^2_{\alpha,h}$, for a chi-squared distribution with $h$ degrees of freedom at the $100(1-\alpha)$th percentile. The accuracy of the final ARIMA-GARCH model and the LSTM model was compared using root mean square error(RMSE).

# Chapter Four

## 4 Data Analysis

### 4.1 Introduction

This chapter presents the data analysis and interpretation which draws from the objectives of the study. The main tasks of this study are: Establish the best traditional forecasting method for financial data and compare traditional methods with DNN. This was done by comparing the performances of ARIMA-GARCH and then after use the models to measure the performance of DNN model. In order to solve the task at hand, the practical part of dissertation was performed in the following way: In the first step, we tested three different models for time series forecasting: ARIMA, GARCH+ARIMA and DCNN. The effectiveness of DNN was compared against that of GARCH+ARIMA. For this purpose we selected two financial data sets from two different markets.

### 4.2 Data Overview

The data sets used in this section are S&P500 US Equity Index and FTSE100 index of the largest UK companies by market cap. The data extracted was for the period between January 2007 to January 2019 to ensure that all financial regimes were captured. We decided to use data from UK and US financial markets because of the trading frequency in these markets which is unmatched in any other parts of the globe.

### 4.3 Analysis Using ARIMA models

In this section we fitted ARIMA models to SP500 us Equity Index and FTSE100. We used the **forecast** library, written by Rob J Hyndman to make the forecasting. The realization of GSPC returns is shown in Figure2 From the figure we deduced that the time series experiences periods of volatility clustering. It's evident from the chart that volatility is not stationary in time. This is clearly depicted from the correlogram plot, given in Figure3.

We saw a number of peaks, at k=1 and k=5, these are definitely statistically significant and goes beyond a white noise model. We had evidence of long-memory processes as there are statistically

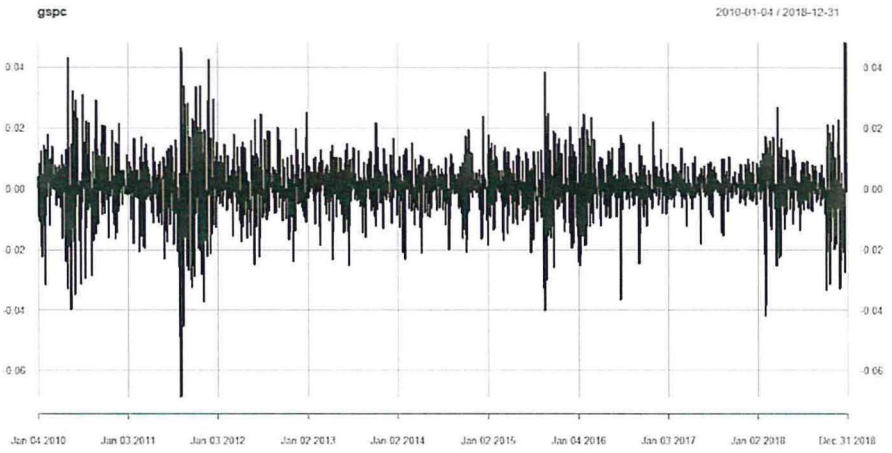


Figure 2: 1st Order Difference Daily Logarithmic Returns of S&P500 Close Prices

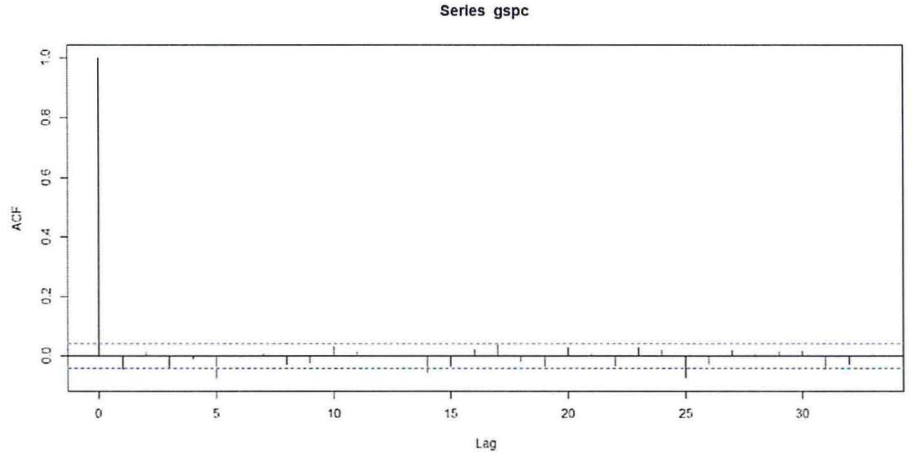


Figure 3: Correlogram of First Order Difference Daily Log Returns of S&P500 Closing Prices

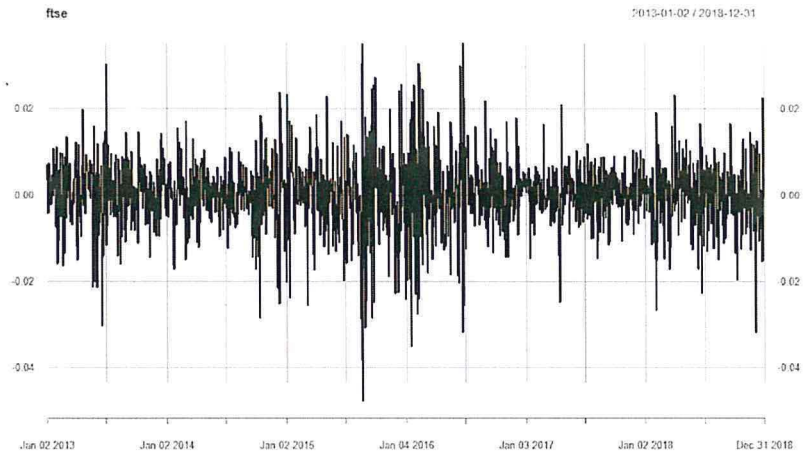


Figure 4: First Order Difference Daily Log Returns of FTSE100 Closing Prices

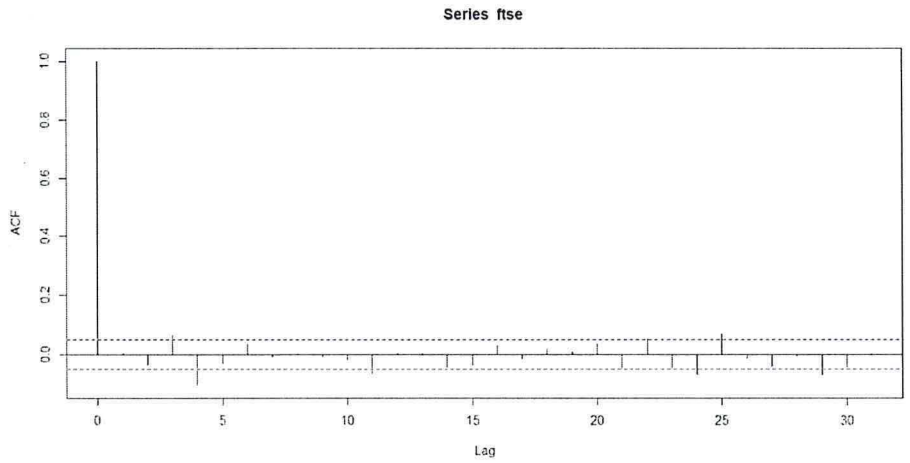


Figure 5: Correlogram of First Order Difference Daily Logarithmic Returns of FTSE100

significant peaks at k=14, k=17 and k=25. We repeated the same for the FTSE100 data, the plot we obtained is as shown in Figure4 Again we could see once more that the volatility isn't stationary in time. The correlogram showed significant peaks at k=4 and k=11 plus evidence of long-memory at k=24, k=25 and k=29. This is shown in Figure 5 We fitted an ARIMA model to see if it would explain some or all of these effects.To carry out this, we looped through the combinations of $p$, $d$ and $q$, to find the optimal ARIMA(p,d,q) model, that is, the order combination that minimizes the Akaike Information Criteria (AIC), we found that ARIMA of order $p = 4$, $d = 0$, $q = 4$ was selected. Note that $d = 0$, since we had taken the first order difference.We plotted the correlogram of the residuals to see if we had any traces of discrete white noise, see Figure 6 There were significant peaks at $k = 17$ and $k = 25$, given that we should expected to see statistically significant peaks simply due to sampling variation $5\%$ of the time. We performed a Ljung-Box test to see if we had a good fit. We

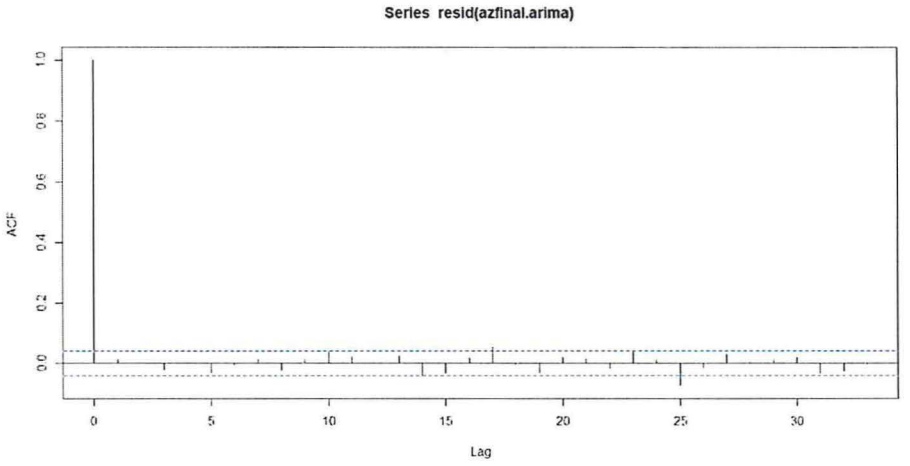


Figure 6: Correlogram of residuals of ARIMA(4,0,4) model fitted to SP500 rnd of day

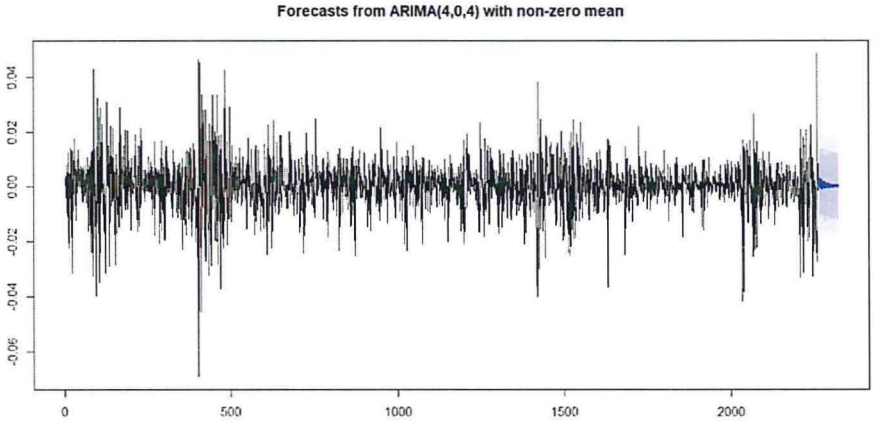


Figure 7: 60-day forecast of S&P500 daily log returns

found a p-value of 0.1176 which is greater than 0.05 and so we had evidence of a good fit at 95% level. Next we forecasted the 60 days ahead for the S&P500 series, this is shown in Figure7 The point forecast for the next 60 days with 95%(dark blue) and 99% (light blue) error bands.

We carried out the same procedure for the FTSE100, then we found out that AIC showed that the best model is ARIMA (3,0,4) model, notice that $d = 0$ because we had already taken the first order differences of the series. We plotted the residuals of the fitted model to see if we have evidence of discrete white noise, see Figure 8 The correlogram looked promising, we carried out Ljung-Box test for the FTSE500 to confirm that it was a great fit. This gave us a p-value 0.6235 which is greater than 0.05, this gave us the evidence of a great fit.

From the two analyses we see that the ARIMA model worked best because of the absence of excessive volatility clustering. This impacts the autocorrelation of the series in question in the sense

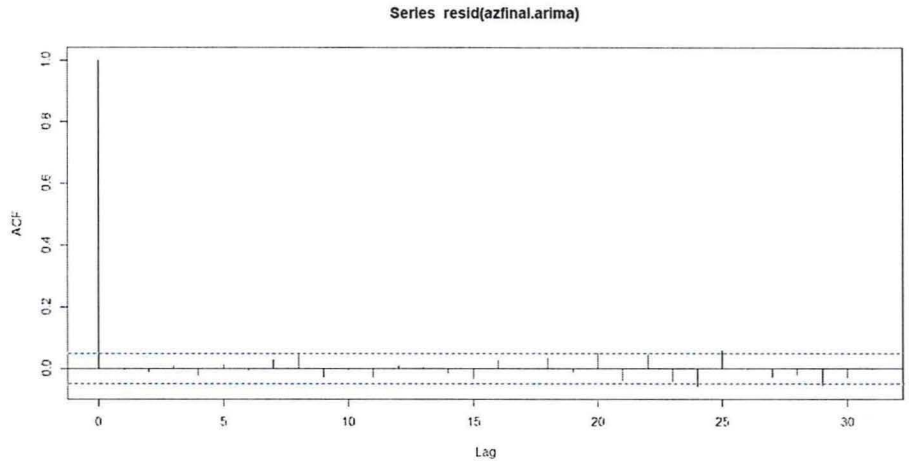


Figure 8: Correlogram of residuals of ARIMA(3,0,4) model fitted to FTSE100 end of day returns

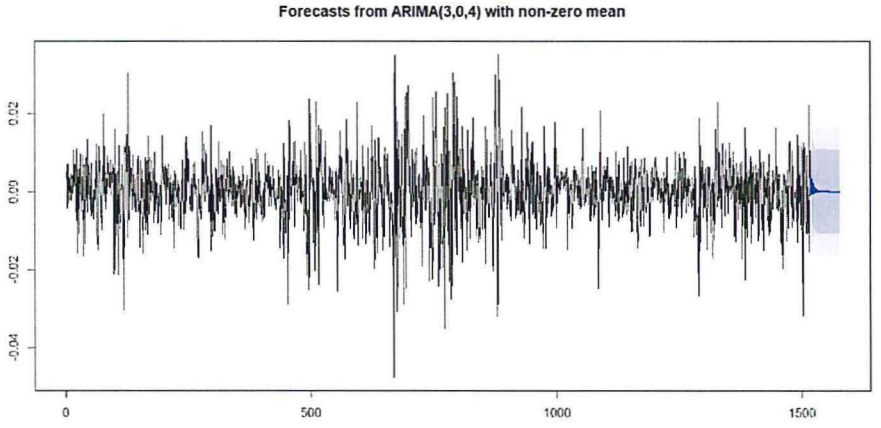


Figure 9: Figure 4.8: 60-day forecast of FTSE100 end of daylog returns

that it makes the series appeared to be stationary than it as been in the past.We plotted the forecast of the FTSE100 daily log returns as shown in Figure9 next we carried out the analyses of the two indexes using the GARCH models and see how it's going to perform.

## 4.4 Analyses Using ARIMA-GARCH models

In this section we took a look at the ARIMA-GARCH, we used this to explain more of the autocorrelation in indexes series. We fitted ARIMA and GARCH to the S&P500 and FTSE100 index. We plotted the values for the FTSE100 as shown in figure10 We saw the evidence that there are periods of increased volatility, around 2008 and 2009, 2012, 2015 and towards the end of the 2018. We removed the **NA** generated by the differencing procedure, then we fixed the possible ARIMA(p,d,q)

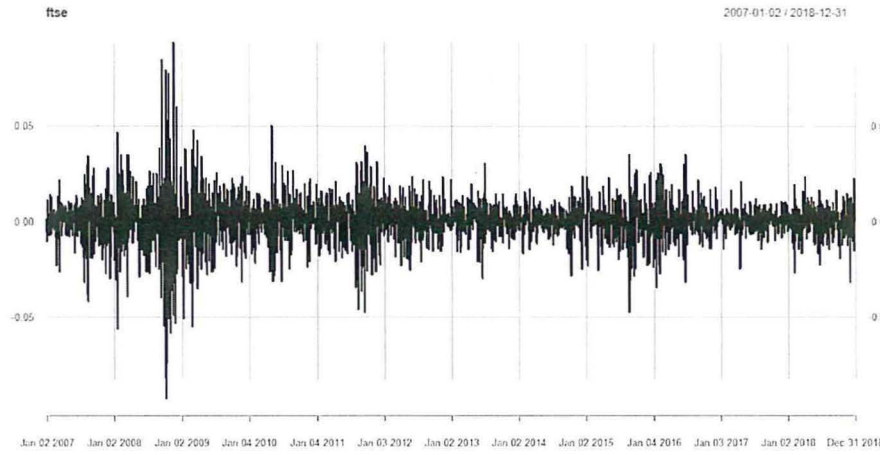


Figure 10: Differenced Log returns of the daily closing prices of the FTSE100 stock index

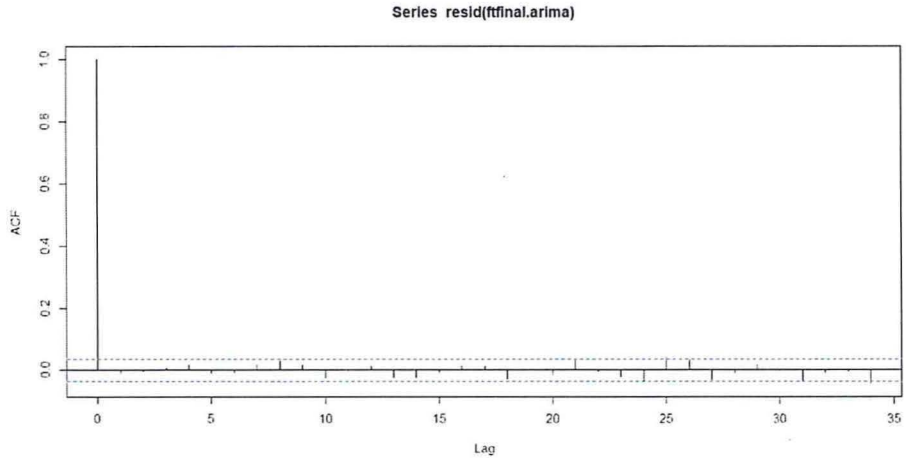


Figure 11: Residuals of an ARIMA(4,0,4) fit to the FTSE100 diff lo returns

model. Since we had differenced the GSPC returns we expected the $d$ component to be equal to zero. Thus we received a final order of ARIMA(4,0,4) model.This implies that the model's order has four autoregressive paramaters and four moving average parameters. Next we explored if the residuals of the model fit possess evidence of conditional heteroskedastic behaviour. Here we used the correlogram of the residuals, as shown in figure 11 This looks like the realization of a white noise process indicating that we managed to achieve the a great fit with the ARIMA(4,0,4) model. We tested for conditional heteroskedatsicity, we did this by squaring the residuals first and plotting the corresponding correlogram, this is seen in figure12 From the figure we can see that there is a clear evidence of serial correlation in the squared residuals, we therefore concluded that conditional heteroskedastic behaviour is present in the differenced log returns of the series of the FTSE100. Hence we had to fit a GARCH model, with used **Trace=F** parameter in R to redact the outputs which were excessive. To test for the goodness of fit, we plotted the correlogram of the GARCH residuals and the square

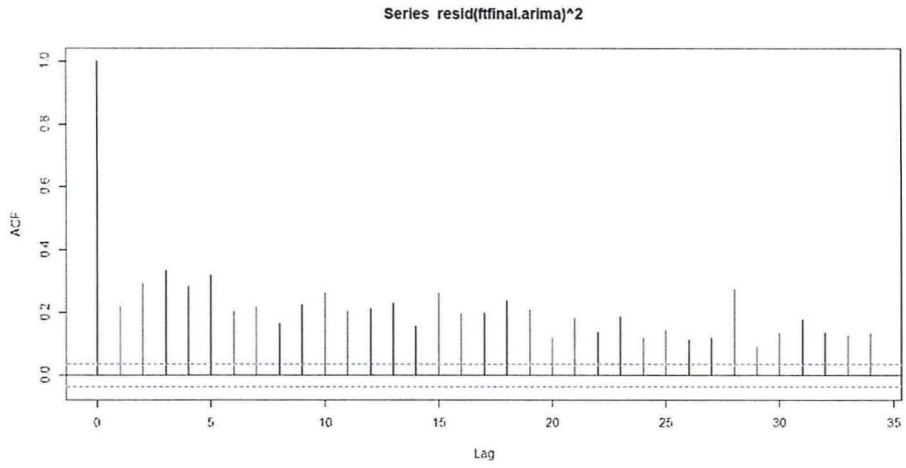


Figure 12: Squared residuals of an ARIMA(4,0,4) FIT TO FTSE100 diff log returns

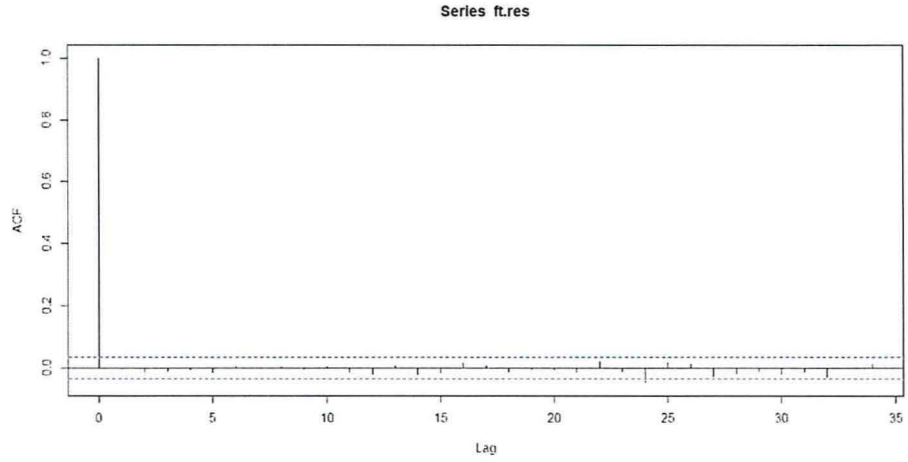


Figure 13: Residuals of GARCH(p,q) fit to the ARIMA(4,0,4) fit of the FTSE100 diff lo returns

GARCH residuals as shown the Figure13 The correlogram looks like a realization of the white noise process indicating a fairly good fit. Next we tried to fit the squared residuals as shown in Figure14

From the figure we could see what looked like the realization of a discrete white noise process, this was an indication that we have 'explained' the autocorrelation present in the squared residuals with a hybrid mixture of two models, that is ARIMA(p,d,q) and GARCH(p,q). We performed the same procedure for the SP500. We got the log differences and remove the **Na** values the fitted the appropriate ARIMA(p,d,q) model. We obtained the following: From 15 as we have seen in the previous example that there are periods of increased volatility, particularly in the periods around 2008-2009, early 2012, late 2015, early and late 2018. We obtained an ARIMA(3,0,3) model. The value of $d$ was zero because we had already differenced the S&P500 returns once. We also tested if the residuals of this model had evidence of conditional heteroskedasticity. We plotted the correlogram of the residuals,

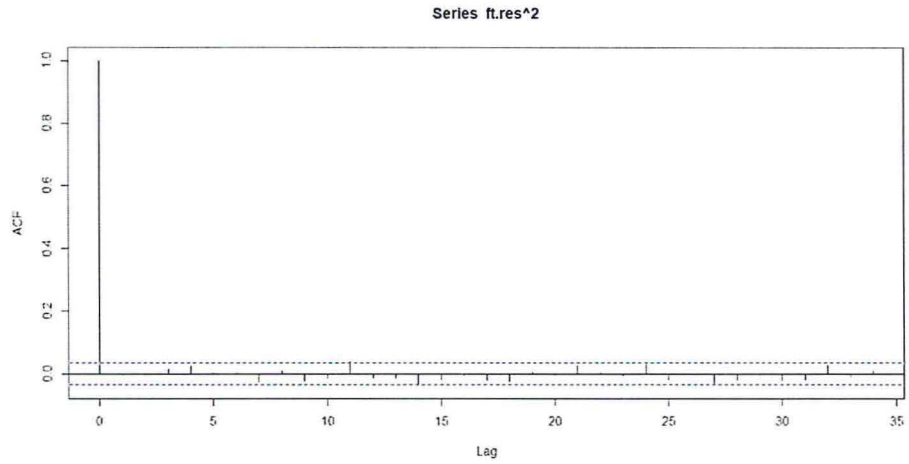


Figure 14: Squared residuals of a GARCH(p,q) fit to the ARIMA(4,0,4) fit of the FTSE100 diff log returns

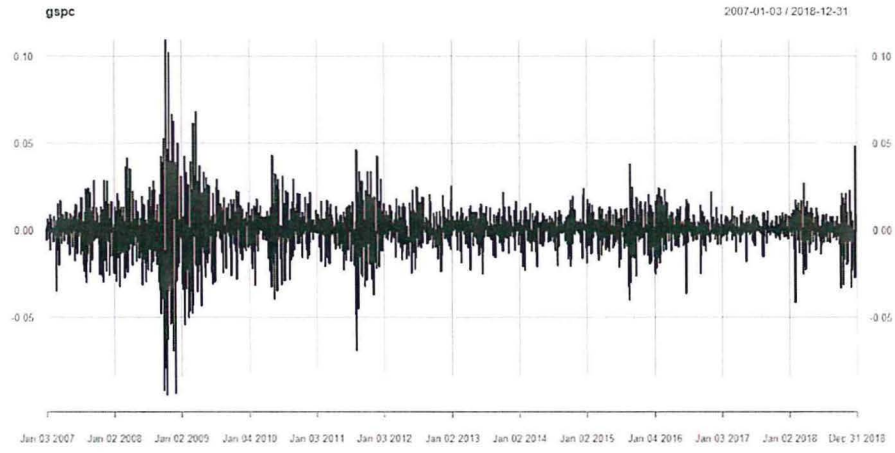


Figure 15: Differenced log returns of daily closing price of the S&P500 stock index

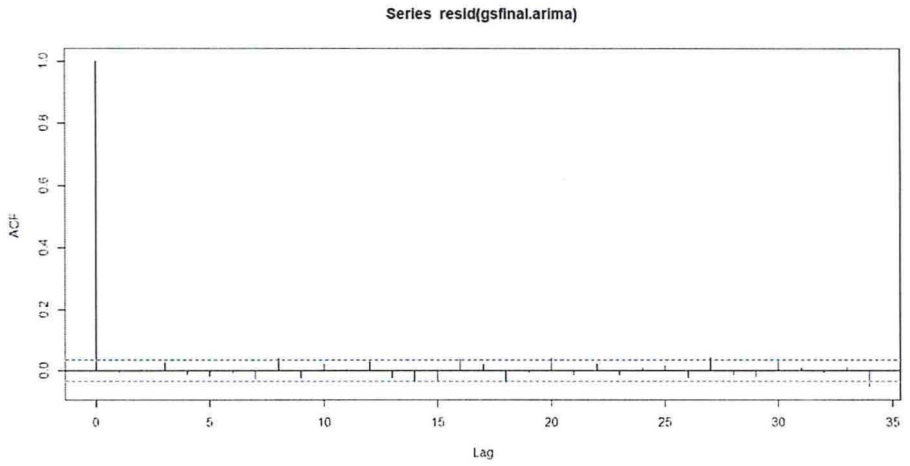


Figure 16: Residuals of ARIMA(3,0,3) fit to S&P500 diff log returns

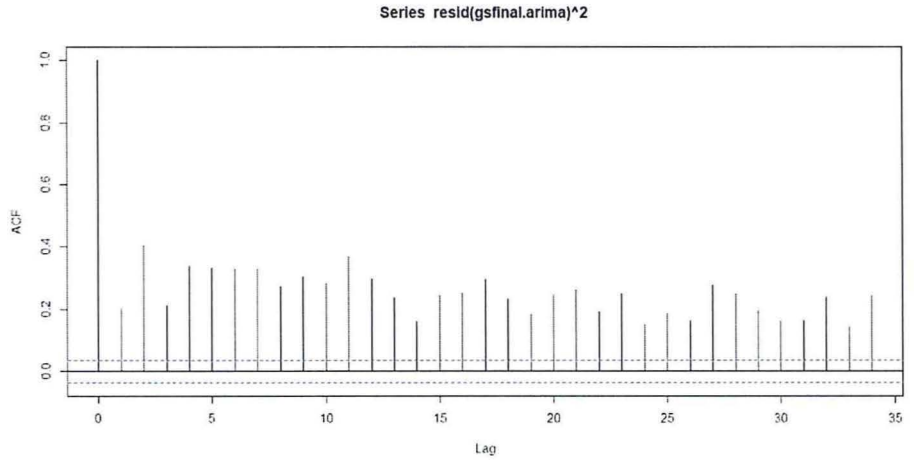


Figure 17: Squared residuals of ARIMA(3,0,3) fit to S&P500 diff log returns

which looked like a realization of the white noise process, the results are shown in Figure16 Next we plotted the correlogram of the squared residuals to test for conditional heteroskedastic. We obtained evidence of serial correlation in the squared residuals in the diff log returns of S&P500 index, as shown in 17. Next we fitted the GARCH model as we did before with FTSE100 series, we plotted the correlogram of the GARCH residuals and the square of the GARCH residuals as shown in 18. We were able to obtain once more what like realization of the discrete white noise process indicating an excellent fit, next we tried the squared residuals as shown in the correlogram in figure 19. Once again we managed to obtain what looked like the realization of the discrete white noise, this implied that we had tried to capture the serial correlation present in the squared residuals with the mixture of the two models. The final plot of the model is indicated in 21.

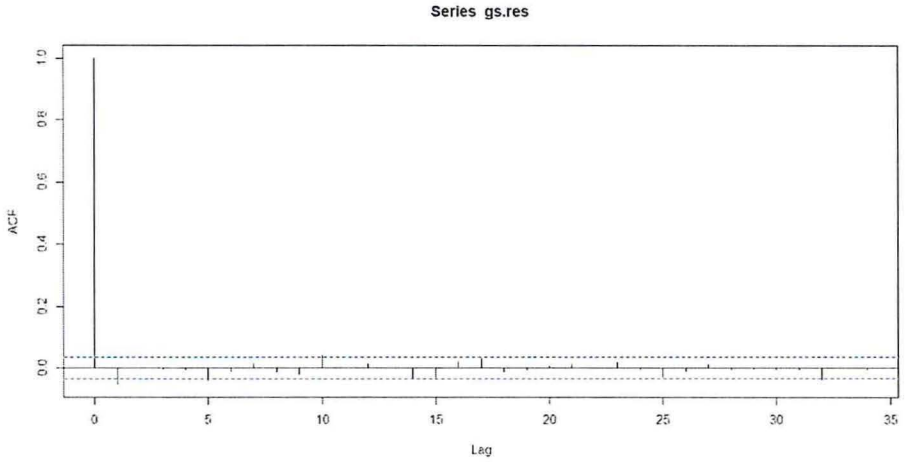


Figure 18: Residuals of GARCH(p,q) fit to ARIMA(3,0,3) fit of the S&P500 diff log returns

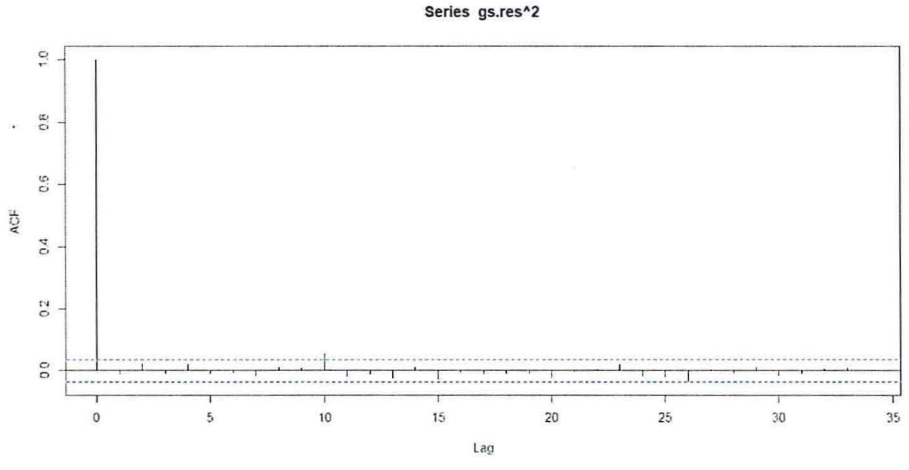


Figure 19: Squared residuals of GARCH(p,q) fit to the ARIMA(3,0,3) fit of the SP500 diff log returns

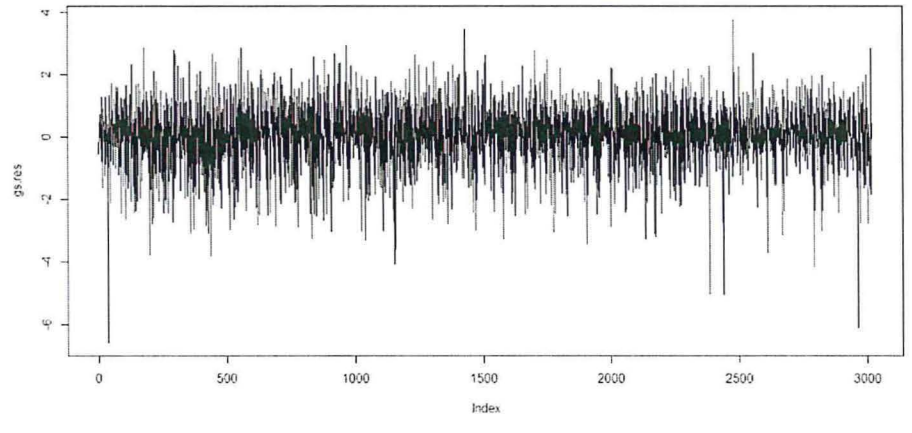


Figure 20: Plot of Final ARIMA-GARCH Model for GSPC

### 4.5 Time Series Prediction Using LSTM Deep Neural Networks

In financial time series prediction we do not assume specific static function which can be mapped since the market is assumed to follow a random walk. Thus, the movements are purely stochastic, this means that it is a pure random walk process and has no predictable patterns thus attempting to model it would be pointless. But basing from the fact that the studies have shown that the stock market might not be purely stochastic process, this allowed us to theorize that the time series may well have some kind of hidden pattern. The hidden pattern is the serial correlation between random variables. We used the LSTM deep networks to predict these hidden patterns.

As the previous analysis, we used the S& P 500 data. The data we have been using so far contains the Open, High, Low, Close Prices plus the volume of the equity indexes. We shifted the data to start from Jan, 2018 to Jan 2019, so that we can see the model can capture all the financial trends including the extreme periods.

We transformed the data to stationary by differencing, in this case we got the log of the returns too since we were focusing on the volatility. Another reason for differencing was to remove the components of the data that were dependent on time so as to increase the predictive power of the model. To implement LSTM, the data is required to be in supervised learning mode, by this we mean having a target variable Y and predictor X, to achieve this we transformed the series by lagging the series and having the value as time $(t - k)$ as input and the value at time $t$ as the output, for a k-step lagged dataset.

Next we split the data into training and testing set. Unlike in most analysis where the training and testing data sets are randomly sampled, with financial time series data the order of the observation does matter. We split the data such that first 70% of the series was training set and 30% was the testing set.

As neural network model requires, we rescaled the input data X to the range of the activation function, since we used sigmoid function, the range was [-1,1], see 21. This ensures that the max and min values of the data do not influence the model.

After this we reverted the predicted values to the original. We compiled the the model, specifying the mean squared error as the loss function, the optimizing algorithm was set to Adaptive Monument Estimation (ADAM), we also used accuracy to asses the model performance.

Finally we fitted the model, we set the argument *shuffle*=FALSE to avoid shuffling of the training set and maintaining the dependencies between $x_i$ and $x_{i+t}$. LSTM also required resetting of

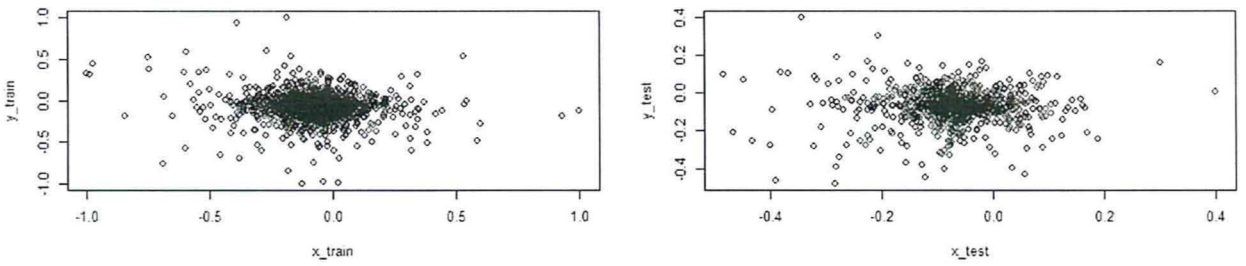


Figure 21: Normalized Data of Train and Test Set

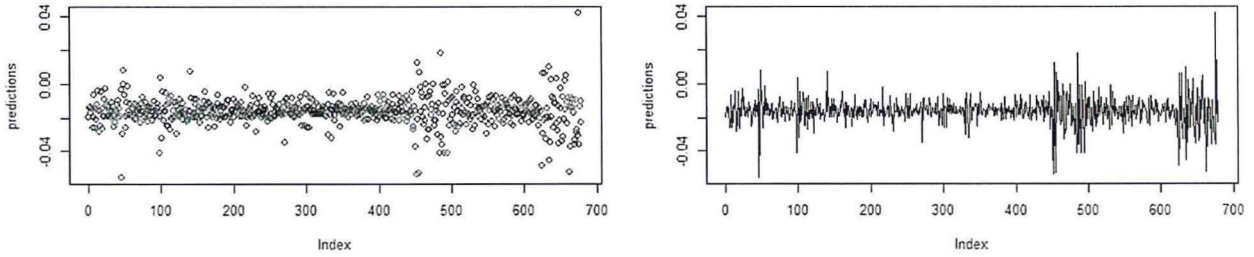


Figure 22: Predicted Values of the LSTM Model

the network state after each epoch. To achieve this we ad to loop over epochs where within each epoch we had to fit the model and reset the states. Then we predicted the the values as shown in 22 We found out that with addition of dimensions, we were able to improve the output of the prediction.Not only were the prevailing trend from the start and the accuracy of the trend lines seems to improve but the predictor trend lines seem to have more accuracy in predicting the upcoming dips.

# Chapter Five

# 5 Discussion

## 5.1 Introduction

In this chapter we are going to present the finding from the the previous chapter, we are going to discuss the findings based on the objectives of the study as stated; which was application of traditional time series tools as a benchmark of evaluating deep neural networks in machine learning performance in time series forecasting. Thus use traditional forecasting tools to evaluate the effectiveness of deep neural networks as a tool for forecasting financial time series. Lastly, for LSTM model we trained the deep neural networks using tensorflow Keras to come up with a predictive model for the time series then compared the performances with the original data and lastly we saw how LSTM was performing compared to the ARIMA-GARCH Model.

## 5.2 Findings

### 5.2.1 Analysis Using ARIMA models

We fitted the ARIMA models to S&P500 US equity Index and FTSE100 index of the largest UK companies by market capitalization. From the realizations of the returns as was indicated in 2 and 4 we show that the time series of FTSE100 and S&P500 experiences periods of volatility clustering, see 23 We could also see from 23 that the volatility is not stationary in time, this was clearly depicted from the correlogram of the plots. In the case of S&P500 we could see many peaks, including k=1 and k=5, see 3 this were statistically significant which suggested that a white noise model could not capture such dependencies. We repeated the same experiment for FTSE100 financial data, we plotted the return realizations and the correlogram, both showed the series was experiencing periods of volatility clustering. From the correlogram plot of the FTSE100, we could see significant picks at k=4 and k=11 plus evidences of long-memory at k=24, k=25 and k=29, see 24 This further suggested that a white noise model could not capture the dependencies in these series. Both findings hinted that we could not capture the dependencies in the model using white noise, we applied the ARIMA model on the dataset. Our aim was to find an ARIMA(p,d,q) model with a combination $p, d$ and $q$

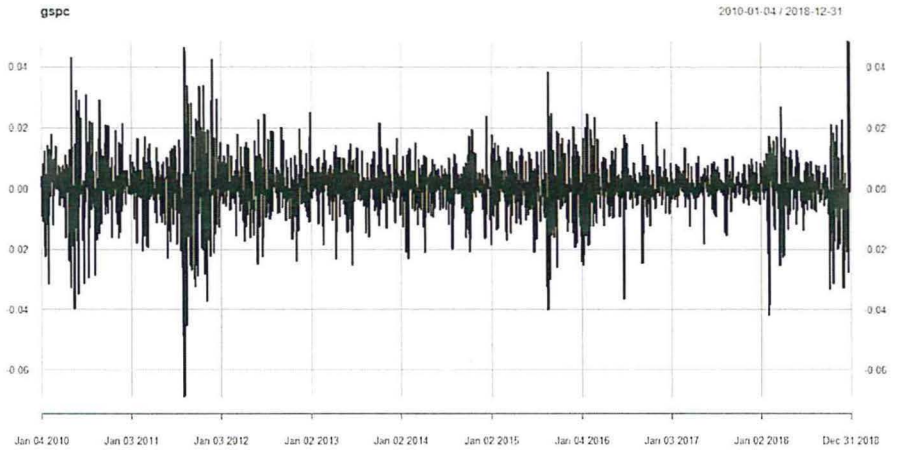


Figure 23: First Order Difference Daily Logarithmic Returns of S&P500 Closing Prices

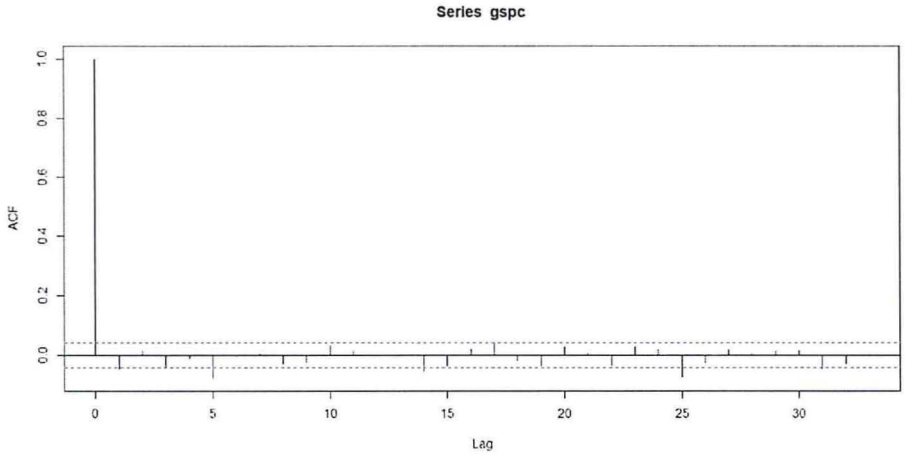


Figure 24: Correlogram of First Order Difference Daily Logarithmic Returns of S&P500 Closing Prices

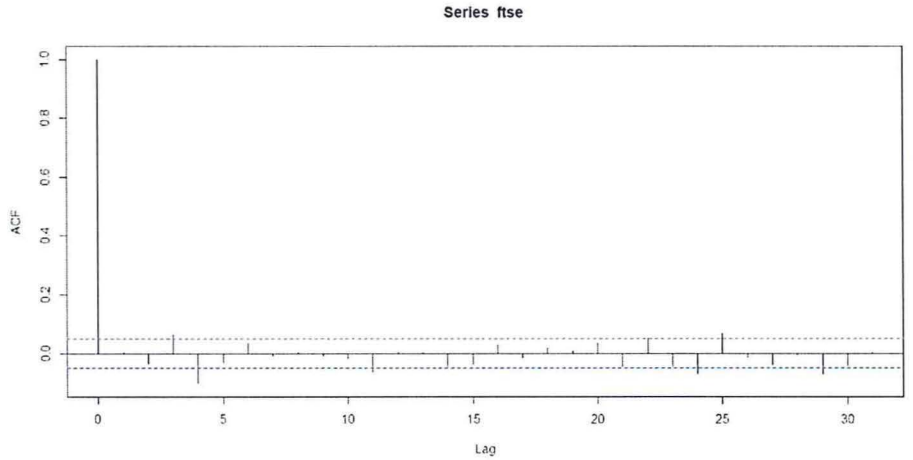


Figure 25: Correlogram of First Order Difference Daily Logarithmic Returns of FTSE100

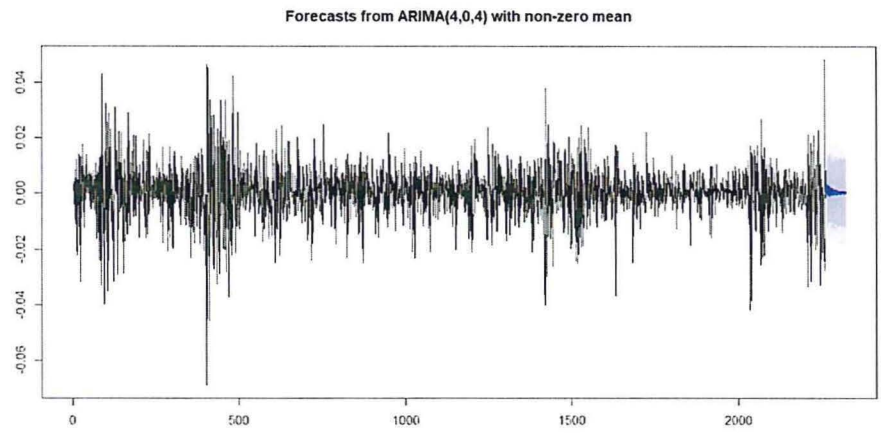


Figure 26: 60-day forecast of S&P500 daily log returns

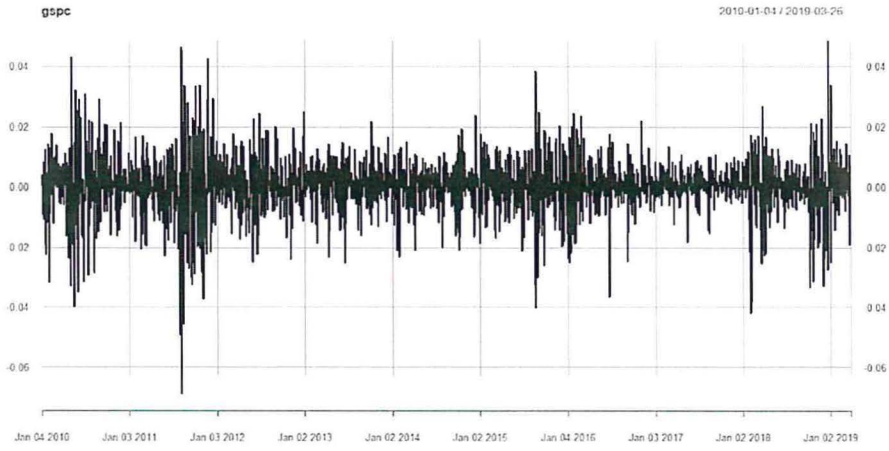


Figure 27: Actual S&P500 daily log returns

that minimizes the Akaike Information Criteria. We looped through the amalgamation of $p, d$ and $q$ to find the optimal ARIMA(p,d,q), for S&P500 we found that ARIMA(4,0,4) was selected. Note that $d = 0$ because of the first difference. We also performed a Ljung-Box test and we found the value of X-squared to be 27.663, p-value of 0.1176, which was greater than 0.05 indicating evidence of a good test at $95\%$ level. From the findings we also saw significant peaks at $k = 17$ and $k = 25$, as shown in the correlogram, which hinted that we are not badly off since it is expected to see statistically significant peaks 5% of the time due to sampling variation. We performed a 60 days forecast for the model, the data was running from '2010-01-01'to '2019-01-01'. The results were as shown in 26 We then compared it with the actual results as shown in 27. The results were quite impressive, as the actual results were within the 99% error bands of the forecast results. We carried out the same procedure for the FTSE100, we obtained that the best ARIMA(p,d,q) that was able to model FTSE100 was ARIMA(3,0,4), we also performed a Ljung-Box test and we found X-squared value of a p-value

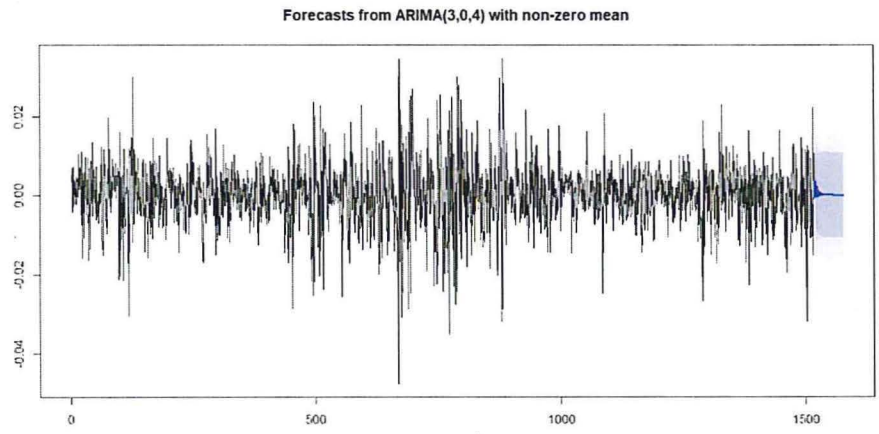


Figure 28: 60-day forecast of FTSE100 daily log returns

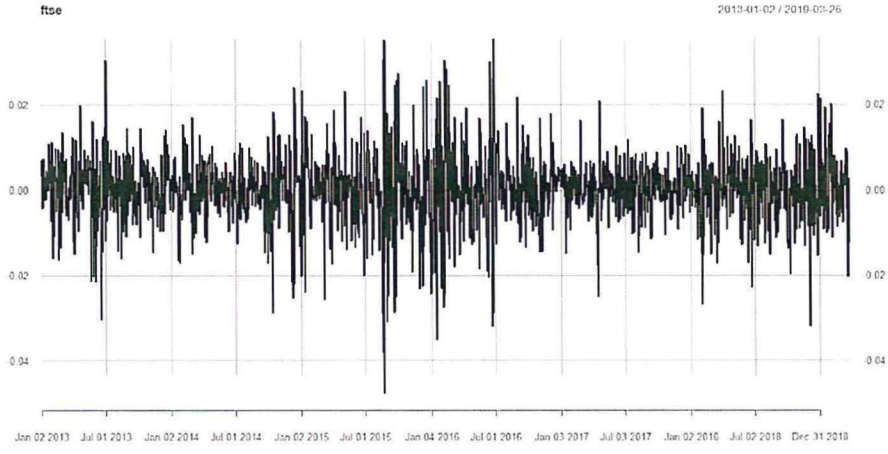


Figure 29: Actual FTSE100 daily log returns

of 0.6235, this is shown in 1, this was impressive and important findings in the understanding of modeling time series, because it hinted that little is know about ARIMA(p,d,q) model in the quantitative finance realm. It is not a one fits all model, as the combinations that worked well for FTSE100 wasn't working for S&P 500. This is seen in the difference see in the combinations used for both series. We also performed a 60 days forecasting, see 28 We compared it with actual outcome, see 29, from the figure we can see that when comparing the actual results against the forecast, we see that the actual result is within the 99% error bands of the forecast result. This indicated that the we had a good model.The results demonstrated that ARIMA model will work best in times of low volatility, but what happens when we have excessive volatility clustering? Our model performed really great because we had not included the periods of excessive volatility, we had to introduce another set of models in order to come up with superior results that can be used in the event of extreme volatility clustering periods.

| Symbol | p-value | X-squared | df |
|---|---|---|---|
| GSPC | 0.1176 | 27.663 | 20 |
| FTSE | 0.6235 | 17.451 | 20 |

Table 1: ARIMA Statistical Analysis

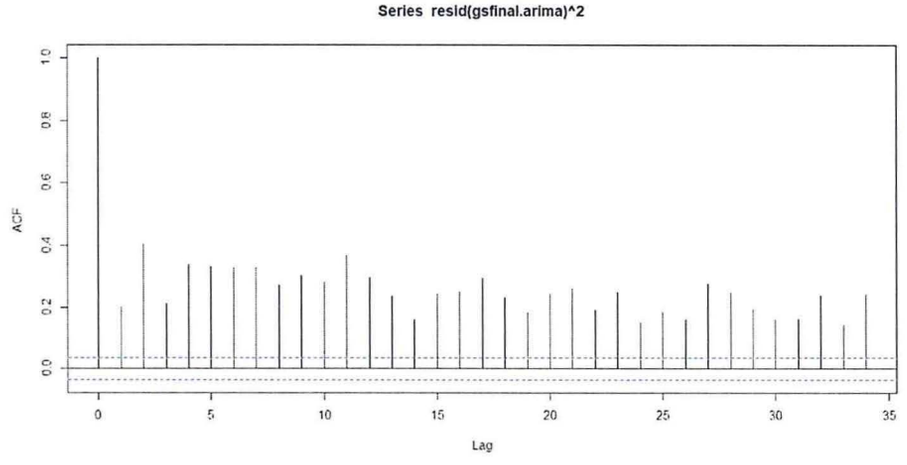


Figure 30: Squared residuals of ARIMA(3,0,3) fit to S&P500 diff log returns

### 5.2.2 Analyses Using ARIMA-GARCH

We used the GARCH model to explain more autocorrelation properties that had not been explained by the ARIMA model. In the data we incorporated the periods of increased volatility which was around 2008, 2009, 2012, 2015 and towards the end of 2018. We fitted ARIMA model on S&P500 and we obtained ARIMA(4,0,4) as the best fit, we expected the $d$ component to be equal to zero because we had differenced the returns. Next we explored if the model's residuals had evidence of conditional heteroskedasticity, see 11, the result looked like the realization of the discrete white noise which indicated that ARIMA(4,0,4) is a great fit. Next we tested the conditional heteroskedastic behaviour by squaring the corresponding correlogram, see 30 This analyses gave us evidence of serial correlation in the diff log returns of the S&P500 index. We fitted the GARCH model, the results cast new light on the next we fitted the GARCH model on the series and we plotted the correlogram to see if the model had explained out all the serial correlation. The results are as shown in 31 From the graph we see that we had obtained what looked like the realization of the discrete white noise, which indicated that the fit was relatively great, this means that the we had captured the autocorrelation present when we squared residuals of the mixture of ARIMA and GARCH models. We repeated the same for the FTSE100 and we obtained the best ARIMA model was ARIMA(4,0,4) after we plotted the square residuals we got no evidence of serial correlation then fitted the GARCH model and result

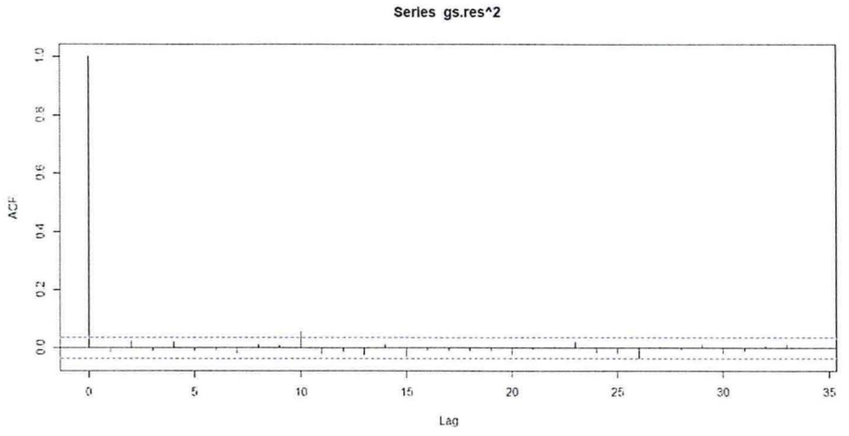


Figure 31: Squared residuals of GARCH(p,q) fit to the ARIMA(3,0,3) fit of the SP500 diff log returns

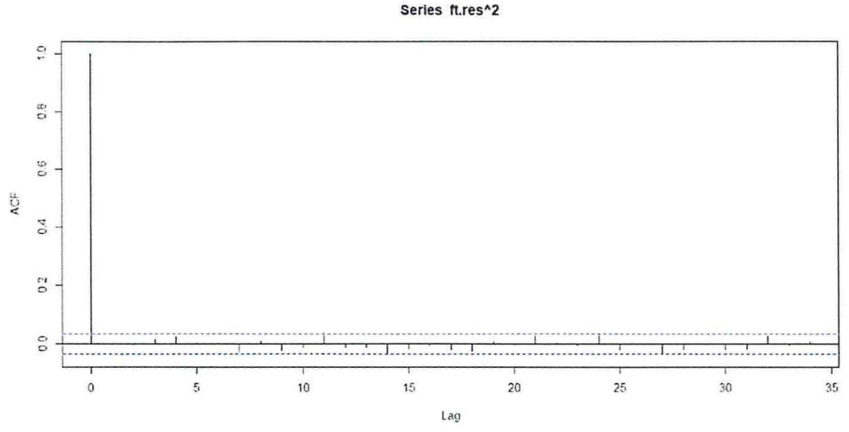


Figure 32: Squared residuals of a GARCH(p,q) fit to the ARIMA(4,0,4) fit of the FTSE100 diff log returns

we obtained is seen in 32, which indicated a good fit.

We performed a Box-Ljung test on the model and we obtained X-squared to be 35.127 and the p-value of 0.01944 which is less than 0.05, this shed new light that maybe we might have overfitted the model. We went ahead and calculated the residual mean squared error (RMSE) which we obtained to be 0.9989187, which was way above the LSTM model that we were comparing with, see 2

### 5.3 Time Series Prediction Using LSTM Deep Neural Networks

We used the LSTM deep networks to predict the serial correlation between the random points in the series. 3 shows the summary of the model. To manipulate the power of this modeling technique we analyzed the data starting from Jan 2000 to September 2018. The aim was to capture all the

| Symbol | p-value | X-squared | df | RMSE |
|---|---|---|---|---|
| GSPC | 0.01944 | 35.127 | 20 | 0.99892 |

Table 2: ARIMA-GARCH statistical analysis

| Layer(type) | Output Shape | Param |
|---|---|---|
| hline lstm_1(LSTM) | (1,1) | 12 |
| dense_1 (Dense) | (1,1) | 2 |
| Total params:14 | NA | NA |
| Trainable params:14 | NA | NA |
| Non-trainable params:0 | NA | NA |

Table 3: LSTM Model Summary

financial trends including the extreme periods that has occurred in 2008, 2009, 2012, 2015 and 2018. We created a single dimension model that used Close prices only as the previous models. We ran the model using the return and we compared the performance of the model and how it was able to replicate the initial data set. The point was to evaluate if LSTM can model volatility in stock prices and the accuracy of the model in modeling volatility. 33 The model that we trained was used to perform predictions on the tested data set. Results of the predictions are shown in 34. It is very clear that the model was able to capture the volatility patterns of the true data, though not accurately. The results demonstrated that the resultant network need not know about the time series itself other than the next, this present studies confirmed the findings about (Hatami et al., 2018) in that if the prediction for the initial point was wrong in the network then there's likelihood of the next prediction factoring in the true history and disregard the incorrect prediction, and thus allowing error to be made in the network. We created a full sequence prediction using the hyperparameters we had created in the

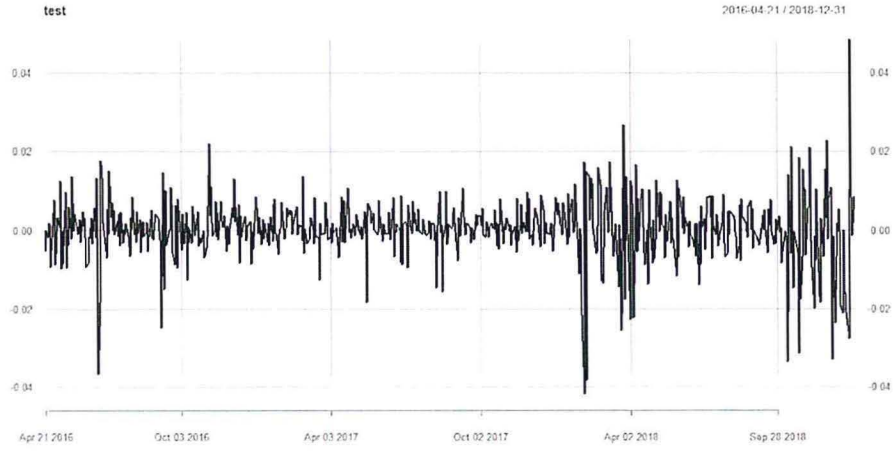


Figure 33: First Order Difference Daily Logarithmic Returns of SP500 Closing Prices

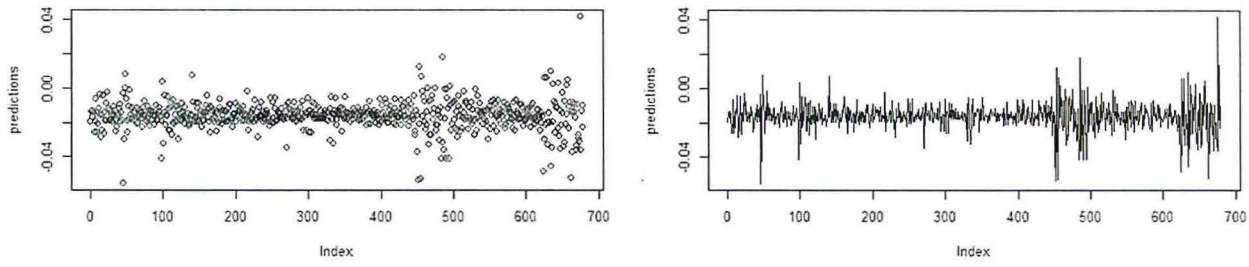


Figure 34: S&P 500 Full Sequence Prediction

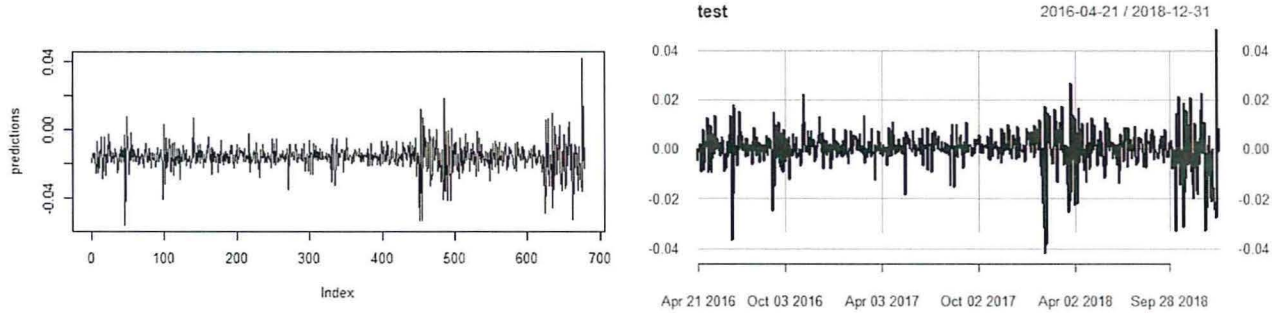


Figure 35: Prediction vs Tested Return series

model above. This led us into another novel finding whereby we found that the model managed to replicate the trained data sets, this showed us the high predictive power that the LSTM. 14 shows the return series of the test data sets and the predicted values, we can see that the predicted series is very close to the tested set. The model had a root mean square error of 0.0178,

## 5.4 Comparison With Previous Works

The studies demonstrated in this chapter match the state of the art model that are currently being used in the financial sector for portfolio construction and algorithmic trading strategies, (Fischer & Krauss, 2018). Here we were comparing the results of the results of the proposed methodology versus the ones that been used traditionally. These results go beyond the previous reports, showing that machine learning models can be used to evaluate, predict and forecast the time series data complexities such as hidden patterns and trends, (Karim, Majumdar, Darabi, & Chen, 2018).This information is very useful especially in the areas such as volatility forecasting, which entails forecasting a period of high/low volatility in the markets.

When we differentiated the statistical evaluation of the ARIMA-GARCH hybrid model and that of the LSTM using the RMSE, we that the LSTM had the lowest RMSE of 0.0178 which was quite low compared to that of ARIMA-GARCH model which had RMSE of 0.9982 and a very low

p-value that failed to outperform that of the ARIMA despite the fact the model showed a very good outcome when we plotted the correlogram.

## 5.5 Limitations of The Study

Training LSTM networks require high computation we therefore decided not to investigate its performance when trained on intraday data such as tick, 1 sec, 1 min or 5 min Open High Close Low data which would have resulted to smoother and quality results. LSTM models require large amounts of data to train, however, this study did not use very massive volumes of data which might have affected the models performance.

Althogh LSTM is widely accepted, it suffers from limitations associated with overfitting since it involves tweaking of hyperparameters.

# Chapter Six

## 6 Conclusions

### 6.1 Introduction

The main task was to establish the best time series forecasting methods. For us to do that we needed to:

a. Examine current time series forecasting method to determine whether they are adequate

b. Consider current changing trend in financial data analysis

In Examining the current time series, we used ARIMA-GARCH model. We started by using the best of ARIMA models in the linear data, we obtained the best ARIMA data by looping through the amalgam of $p, d$ and $q$ to find the best $ARIMA(p, d, q)$. The best ARIMA was the one that had the least Akaike Information Criteria and we used the Ljung-Box test to calculate the p-vaues. The best ARIMA model for the S&P500 was ARIMA(4,0,4) while the best ARIMA model for FTSE100 was ARIMA(3,0,4). The ARIMA model explained forecasted the returns but when we squared the residuals and plotted the correlogram, the correlogram had significant peaks. This was the evidence that the ARIMA had not explained the autocorrelations in the residuals. Next we fitted the GARCH model to the ARIMA residuals and plotted the correlogram of the residuals and the square residuals. The correlogram plots showed that the GARCH had explained serial correlations, there were no significant peaks on the plots. The results showed that combination of the two models forecast returns that are adjusted to volatility. We used the RMSE to determine the accuracy of the ARIMA-GARCH hybrid model, we obtained the RMSE of 0.9982 with a p-value of 0.01944, this is lower than ideal one which is supposed to be greater than 0.05.

In considering changing trends, we employed the use of deep neural network models in forecasting financial time series data. In this case we decided to use Long-Short Term Memory because it is the recent deep neural network architecture that is more powerful than other networks such as recurrent neural networks(RNN). To determine the accuracy of the LSTM models we used RMSE, the LSTM model gave us RMSE of 0.0178.

Finally, we compared the ARIMA-GARCH models, the RMSE of the LSTM models was 0.0178 while that of the ARIMA-GARCH was 0.9982 with p-value 0.01944 which was very low,

this made us conclude that the ARIMA-GARCH had not explained all the autocorrelation. This was proved by plotting the ARIMA-GARCH model, we compared the plot of ARIMA-GARCH with that of the actual data and the plot of the predicted values of the LSTM. The plots indicated that the LSTM model had replicated that of the actual data.

Given this results, we therefore concluded that the LSTM-DNN is a superior time series forecasting tool for financial data, compared to ARIMA-GARCH model.

## 6.2 Discussion about further improvements

One of the most intuitive improvement of the the forecasting results of the LSTM requires training the LSTM model using the in depth intraday financial data that has trade and quotes. Increase the number of variables gives the network the power to find more explanatory patterns which cannot be obtained only from a single variable such as the Close Prices.

The training data may contain sequences that don't represent a typical time series behaviour, and their presence in the training set contributes to incorrect estimation of the model which might lead to decreasing of the final forecasting performance, this may improved through using large amount of training data sets. Training LSTM on massive volume of the training data sets, improves the performance significantly.

Another option on how to improve the forecasting performance is through a technique known as the 'Consensus forecast'. It involves finding a linear combination of forecasted values from different models such that the overall forecasted error will be minimal.

# References


Adhikari, R., Agrawal, R., & Kant, L. (2013). Pso based neural networks vs. traditional statistical models for seasonal time series forecasting. In *2013 3rd ieee international advance computing conference (iacc)* (pp. 719–725).

Arévalo, A., Niño, J., Hernández, G., & Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks. In *International conference on intelligent computing* (pp. 424–436).

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*.

Cao, L., & Tay, F. E. (2001). Financial forecasting using support vector machines. *Neural Computing & Applications*, *10*(2), 184–192.

Cao, L.-J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks*, *14*(6), 1506–1518.

Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues.

Crawford, G. W., & Fratantoni, M. C. (2003). Assessing the forecasting performance of regime-switching, arima and garch models of house prices. *Real Estate Economics*, *31*(2), 223–243.

Dasgupta, C. G., Dispensa, G. S., & Ghose, S. (1994). Comparing the predictive performance of a neural network model with some traditional market response models. *International Journal of Forecasting*, *10*(2), 235–244.

Dingli, A., & Fournier, K. S. (2017). Financial time series forecasting–a deep learning approach. *Int. J. Mach. Learn. Comput*, *7*(5), 118–122.

Dixon, M., Klabjan, D., & Bang, J. H. (2017). Classification-based financial markets prediction using deep neural networks. *Algorithmic Finance*(Preprint), 1–11.

Doering, J., Fairbank, M., & Markose, S. (2017). Convolutional neural networks applied to high-frequency market microstructure forecasting. In *2017 9th computer science and electronic engineering (ceec)* (pp. 31–36).

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669.

Gamboa, J. C. B. (2017). Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*.

Halls-Moore, M. (2017). *Advanced algorithmic trading*. September.

Hamilton, J. D. (1994). *Time series analysis* (Vol. 2). Princeton university press Princeton, NJ.

Hansen, J. V., & Nelson, R. D. (1997). Neural networks and traditional time series methods: a synergistic combination in state economic forecasts. *IEEE transactions on Neural Networks*, *8*(4), 863–873.

Hatami, N., Gavet, Y., & Debayle, J. (2018). Classification of time-series images using deep convolutional neural networks. In *Tenth international conference on machine vision (icmv 2017)* (Vol. 10696, p. 106960Y).

Karim, F., Majumdar, S., Darabi, H., & Chen, S. (2018). Lstm fully convolutional networks for time series classification. *IEEE Access*, *6*, 1662–1669.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1-2), 307–319.

Mahfoud, S., & Mani, G. (1996). Financial forecasting using genetic algorithms. *Applied artificial intelligence*, *10*(6), 543–566.

Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. *Econometrica: Journal of the Econometric Society*, 347–370.

Rathnayaka, R. K. T., Seneviratna, D., Jianguo, W., & Arumawadu, H. I. (2015). A hybrid statistical approach for

stock market forecasting based on artificial neural network and arima time series models. In *2015 international conference on behavioral, economic and socio-cultural computing (besc)* (pp. 54–60).

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., & Soman, K. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 1643–1647).

Tan, Z., Zhang, J., Wang, J., & Xu, J. (2010). Day-ahead electricity price forecasting using wavelet transform combined with arima and garch models. *Applied Energy*, *87*(11), 3606–3610.

Xiao, Y., Xiao, J., Liu, J., & Wang, S. (2014). A multiscale modeling approach incorporating arima and anns for financial market volatility forecasting. *Journal of Systems Science and Complexity*, *27*(1), 225–236.

Yaziz, S., Azizan, N., Zakaria, R., & Ahmad, M. (2013). The performance of hybrid arima-garch modeling in forecasting gold price. In *20th international congress on modelling and simulation, adelaide* (pp. 1–6).

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, *14*(1), 35–62.

# Appendix

## R Codes

## Packages Utilized

### 6.2.1 Time Series Analysis

```
library(tseries)
    library(forecast)
    library(quantmod)
    Downoloading and Preparing Data
    getSymbols('ĜSPC', from='2010-01-01', to='2019-01-01')
    getSymbols("F̂TSE", src = 'yahoo', from='2013-01-01', to='2019-01-01')
    saveRDS(GSPC, file = 'GSPC.rds')
    GSPC=readRDS('GSPC.rds')
    saveRDS(FTSE, file = 'FTSE.rds')
    FTSE=readRDS('FTSE.rds')
    gspc=diff(log(Cl(GSPC)))
    plot(gspc)
    acf(gspc, na.action = na.omit)
    ftse=diff(log(Cl(FTSE)))
    plot(ftse)
    ARIMA Analysis of FTSE
    acf(ftse, na.action = na.omit)
    azfinal.aic¡-Inf
    azfinal.order¡-c(0,0,0)
```

```
for (p in 1:4)for(d in 0:1) for (q in 1:
azcurrent.aic¡-AIC(arima(gspc, order = c(p,d,q)))
if (azcurrent.aic¡azfinal.aic)
azfinal.aic¡-azcurrent.aic
azfinal.order¡-c(p,d,q)
azfinal.arima¡-arima(gspc, order = azfinal.order)
azfinal.order
Ljung-Box Test
Box.test(resid(azfinal.arima), lag=20, type="Ljung-Box")
Plotting Final ARIMA Model
¿acf(resid(azfinal.arima), na.action = na.omit)
Forecast plot(forecast(azfinal.arima, h=25))
Repeat of Process with FTSE Analysis
azfinal.aic¡-Inf
azfinal.order¡-c(0,0,0)
for (p in 1:4) for (d in 0:1) for (q in 1:4)
azcurrent.aic ¡- AIC(arima(ftse, order=c(p, d, q)))
if (azcurrent.aic ¡ azfinal.aic)
azfinal.aic ¡- azcurrent.aic
azfinal.order ¡- c(p, d, q)
azfinal.arima ¡- arima(ftse, order=azfinal.order)
azfinal.order
acf(resid(azfinal.arima), na.action = na.omit)
Box.test(resid(azfinal.arima), lag=20, type="Ljung-Box")
```

GARCH MODELS getSymbols('ĜSPC', from='2007-01-01', to='2019-01-01')

getSymbols('F̂TSE', from='2010-01-01', to='2019-01-01')

ftse=diff(log(Cl(FTSE)))

plot(ftse)

ft¡-as.numeric(ftse)

ft¡-ft[is.na(ft)]

ftfinal.aic ¡- Inf

ft ¡- ft[!is.na(ft)]

for (p in 1:4) for (d in 0:1) for (q in 1:4)

ftcurrent.aic ¡- AIC(arima(ft, order=c(p, d, q)))

if (ftcurrent.aic ¡ ftfinal.aic)

ftfinal.aic ¡- ftcurrent.aic

ftfinal.order ¡- c(p, d, q)

ftfinal.arima ¡- arima(ft, order=ftfinal.order)

ftfinal.order

acf(resid(ftfinal.arima))

acf(resid(ftfinal.arima)2̂)

library(tseries)

ft.res¡-ft.garch$res[-1]

acf(ft.res)

acf(ft.res)

acf(ft.res2̂)

GARCH S&P500

getSymbols('ĜSPC', from='2007-01-01', to='2019-01-01')

```
gspc=diff(log(Cl(GSPC)))

gs¡-as.numeric(gspc)

gs¡-gs[!is.na(gs)]

gsfinal.aic¡-Inf

gsfinal.order¡-c(0,0,0)

for (p in 1:4) for (d in 0:1) for (q in 1:4)

gscurrent.aic ¡- AIC(arima(gs, order=c(p, d, q)))

if (gscurrent.aic ¡ gsfinal.aic)

gsfinal.aic ¡- gscurrent.aic

gsfinal.order ¡- c(p, d, q)

gsfinal.arima ¡- arima(gs, order=gsfinal.order)

gsfinal.order

acf(resid(gsfinal.arima))

acf(resid(gsfinal.arima)2̂)

gs.garch ¡- garch(gs, trace=F)

gs.res ¡- gs.garch$res[-1]

plot(gs.res, type = 'l')

acf(gs.res)
```

Box-Ljung test

```
Box.test(resid(gs.garch), lag=20, type="Ljung-Box")
```

Calculating RMSE IN GARCH

GSPC

```
data¡-read.csv(choose.files(), header = TRUE)

head(data)
```

```
plot(data[,8])

plot(data[,8], type = 'l') GSPC returns

plot(gs.res, type = 'l') plotting GARCH Model

ret¡-data[,8]Picking returns from data

m¡-gs.res

o¡-ret

FUNCTION FOR RMSE

RMSE = function(m, o)

sqrt(mean((m - o)^2))

RMSE(gs.res, ret)

LSTM

library(keras)

library(tensorflow)

getSymbols('ĜSPC', from='2007-01-01', to='2019-01-01')

transform data to stationarity

diffed¡-diff(log(Cl(GSPC)))

head(diffed)

lag_transform ¡- function(x, k= 1)

lagged = c(rep(NA, k), x[1:(length(x)-k)])

DF = as.data.frame(cbind(lagged, x))

colnames(DF) ¡- c( paste0('x-', k), 'x')

DF[is.na(DF)] ¡- 0

return(DF)

supervised = lag_transform(diffed, 1)
```

head(supervised)lag_transform ¡- function(x, k= 1)

lagged = c(rep(NA, k), x[1:(length(x)-k)])

DF = as.data.frame(cbind(lagged, x))

colnames(DF) ¡- c( paste0('x-', k), 'x')

DF[is.na(DF)] ¡- 0

return(DF)

supervised = lag_transform(diffed, 1)

head(supervised)

split into train and test sets

N = nrow(supervised)

n = round(N *0.7, digits = 0)

train = supervised[1:n, ]

test = supervised[(n+1):N, ]

scale data

scale_data = function(train, test, feature_range = c(0, 1))

x = train

fr_min = feature$_r ange[1]$

fr_max = feature$_r ange[2]$

std_train = ((x - min(x) ) / (max(x) - min(x) ))

std_test = ((test - min(x) ) / (max(x) - min(x) ))

scaled_train = std_train *(fr_max -fr_min) + fr$_m in$

scaled_test = std$_t est * (fr_max - fr_min) + fr_m in$

return( list(scaled_train = as.vector(scaled$_t rain), scaled_test = as.vector(scaled_t est), scaler = c(min = min(x), max = max(x))))

```
Scaled = scale_data(train, test, c(-1, 1))

y_train = Scaled$scaled_train[, 2]

x_train = Scaled$scaled_train[, 1]

y_test = Scaled$scaled_test[, 2]

x_test = Scaled$scaled_test[, 1]

inverse-transform

invert_scaling = function(scaled, scaler, feature_range = c(0, 1))

min = scaler[1]

max = scaler[2]

t = length(scaled)

mins = feature_range[1]

maxs = feature_range[2]

inverted_dfs = numeric(t)

for( i in 1:t)

X = (scaled[i]- mins)/(maxs - mins)

rawValues = X *(max - min) + min

inverted_dfs[i] ¡- rawValues  return(inverted_dfs)

inverse-transform

invert_scaling = function(scaled, scaler, feature_range = c(0, 1))

min = scaler[1]

max = scaler[2]

t = length(scaled)

mins = feature_range[1]

maxs = feature_range[2]
```

```
inverted_dfs = numeric(t)

for( i in 1:t)

X = (scaled[i]- mins)/(maxs - mins)

rawValues = X *(max - min) + min

inverted_dfs[i] ¡- rawValues

return(inverted_dfs)

model ¡- keras_model_sequential()

model%¿%

layer_lstm(units, batch_input_shape = c(batch_size, X_shape2, X_shape3), stateful= TRUE)%¿%

layer_dense(units = 1)

model %¿% compile(

loss = 'mean_squared_error',

optimizer = optimizer_adam( lr= 0.02, decay = 1e-6 ),

metrics = c('accuracy')

)

summary(model)

Epochs = 50 for(i in 1:Epochs )

model %¿% fit(x_train, y_train, epochs=1, batch_size=batch_size, verbose=1, shuffle=FALSE)

model %¿% reset_states()

L = length(x_test)

scaler = Scaled$scaler

predictions = numeric(L)

for(i in 1:L)

X = x_test[i]
```

```
dim(X) = c(1,1,1)

yhat = model %¿% predict(X, batch_size=batch_size)

invert scaling

yhat = invert_scaling(yhat, scaler, c(-1, 1))

invert differencing

yhat = yhat + diffed[(n+i)]

store

predictions[i] ¡- yhat

par(mfrow=c(2,2))

plot(predictions)

plot(predictions, type='l')

N = nrow(diffed)

n = round(N *0.7, digits = 0)

train = diffed[1:n, ]

test = diffed[(n+1):N, ]

par(mfrow=c(2,2))

plot(test)

plot(predictions, type='l')
```